Multi-Agent Reinforcement Learning for Intrusion Detection: A Case Study and Evaluation

Arturo Servin and Daniel Kudenko

Department of Computer Science, University of York
Heslington, York. YO10 5DD, United Kingdom
{aservin,kudenko}@cs.york.ac.uk

Abstract. In this paper we propose a novel approach to train *Multi-Agent Reinforcement Learning* (MARL) agents to cooperate to detect intrusions in the form of normal and abnormal states in the network. We present an architecture of distributed sensor and decision agents that learn how to identify normal and abnormal states of the network using *Reinforcement Learning* (RL). Sensor agents extract network-state information using tile-coding as a function approximation technique and send communication signals in the form of actions to decision agents. By means of an on line process, sensor and decision agents learn the semantics of the communication actions. In this paper we detail the learning process and the operation of the agent architecture. We also present tests and results of our research work in an intrusion detection case study, using a realistic network simulation where sensor and decision agents learn to identify normal and abnormal states of the network.

1 Introduction

Intrusion Detection Systems (IDS) play an important role in the protection of computer networks and information systems from intruders and attacks. Despite previous research efforts there are still areas where IDS have not satisfied all requirements of modern computer systems. Specifically, *Denial of Service* (DoS) and *Distributed Denial of Service* (DDoS) attacks have received significant attention due to the increased security vulnerabilities in end-user software and bot-nets [12]. A special case of DoS are the *Flooding-Base DoS* and *Flooding-Base DDoS* attacks. These are generally based on a flood of packets with the intention of overfilling the network resources of the victim. It is especially difficult to create a flexible hand-coded IDS for such attacks, and machine learning is a promising avenue to tackle the problem. Due to the distributed nature of this type of attacks and the complexities that involve its detection, we propose a distributed reinforcement learning (RL) approach.

In order to evaluate our technique we explore its use in *Distributed Intrusion Detection Systems* (DIDS). Distributed Intrusion Detection Systems (DIDS) is a group of IDS or sensors coordinated to detect anomalies or intrusions. The system can be homogeneous with every sensor of the same kind and type or

R. Bergmann et al. (Eds.): MATES 2008, LNAI 5244, pp. 159–170, 2008.

heterogeneous with a mixture of types. We build our DIDS approach by training a group of heterogeneous sensor agents that must identify normal and abnormal states of the network resulting from Flood-Base DoS and DDoS. We have used the detection of these attacks to test our learning approach for the following reasons:

- Some researchers [2,19] note that a variety of sensor information is required to detect attacks with high levels of confidence.
- The type of attacks disrupt the operation of the network by modifying state information. Spotting these abnormal states can lead to the detection of a flooding attack.
- The abnormal states are characterised by several factors that are normally present in different part of the network and they are only visible to specific networks devices. To identify these events, it is not possible to use a single device or entity.
- To identify events around the network that are visible to only some type of agents it is necessary to use distributed specialised agents. These agents only have partial observability of the whole environment (network).

In addition to proposing a distributed RL approach for intrusion detection, we adapt and evaluate it in a realisic network simulation using the *ns-2* [20] simulator. In this way, we are able to demonstrate the practical applicability of our approach.

2 Technology Overview

Flood-Base DoS and DDoS attacks change the normal behaviour of the network in different ways and spotting these differences could help us to detect the presence of attacks [14]. The distributed operation of these attacks brings on the opportunity to use a distributed and adaptable platform to detect them. We propose to use an architecture based on MARL agents.

IDS are commonly divided in two functional categories; *Anomaly Intrusion Detection* and *Misuse/Signature Intrusion Detection*. Anomaly IDS states that intrusions are deviations of normal traffic. These systems create profiles of different variables over time to get a usage pattern. The difference between the pattern and the current activity triggers an alarm. The advantage of these systems is that they are capable of detecting unknown attacks, however non-malicious activity that does not match normal behaviour can also trigger the intrusion mechanism. This results in a high rate of *false alarms*. On the other hand misuse or signature intrusion detection systems use rule matching to detect intrusions. These systems compare system activity with specific intrusion rules that are generally hard coded. When the observed activity matches the intrusion pattern an intrusion is detected and an action is executed. The flaw of these systems is that regardless of their accuracy and reliability they lack the ability to detect new types of attacks.

Anomaly Intrusion Detection Systems use a variety of schemes to detect normal user patterns from simple statistical to complex machine learning methods.

Although most of the research on IDS using machine learning has been done under an Anomaly Intrusion Detection approach, recent research work incorporates *Machine Learning* to automatic rule generation on misuse/signature intrusion detection. IDS using machine learning try to learn a function that maps input information into different categories. The learning process can be supervised, unsupervised or reinforced.

In Reinforcement Learning an agent learns to act optimally via observations and feedback from the environment in the form of positive or negative rewards [23]. A widely used RL technique is *Q-learning* [24]. In Q-learning the agent iteratively tries to estimate a Q value function that tells the agent how good it is to perform a specific action in a given state. In Q-learning the agent chooses an action a in any given state s, observes the reward r and the next state s'. Then it updates the estimated Q value denoted by $\hat{Q}$ in Eq. 1.

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + \alpha(r + \gamma\ max_a\ \hat{Q}(s',a')) \tag{1}$$

In order to discover which actions lead to the best rewards over time, the agent needs to *explore* and to *exploit* actions. In our experiments we have used *Boltzmann* or *softmax action selection rules* as the exploration/exploitation strategy. When RL is used in real world applications, it is not feasible to store values for all states individually. To tackle this problem we use Tile Coding, a *function approximation* technique.

MARL has shown promise in solving distributed problems, but there are many challenges to overcome when applying it in a realistic network domain, e.g., feature selection, communication, and synchronisation. In a DIDS architecture we require a large number of distributed agents collecting complex network information and coordinating their action under restricted communication.

3 Agent Architecture

In previous research [17], we used a highly abstract IDS scenario to test how a group of agents learn to interpret and coordinate their action signals to detect normal and abnormal activity. We proposed a hierarchical architecture of agents composed by groups of agents or cells. These cells were composed by *sensor agents* (SA) and *decision agents* (DA). SA collect and analyse state information about the environment. Each SA receives only partial information about the global state of the environment and they map this local state to communications action-signals. These signals are received by the DA and without any previous knowledge it learns their semantics and how to interpret their meaning. In this way, the DA tries to model the local state of cell environment. Then it decides which signal-action to execute to a higher level agent outside the cell or in single cell environments the final action to trigger (in our case study it triggers an alarm to the network operator). To expand the number of agents we used hierarchical multi-cell architectures composed of cells of DAs. In these multi-cell environments each DA inside the cells sends an action-signal to a central DA, which in turn sends an action-signal to a higher level DA. When the top

agent in the hierarchy triggers the appropriate action, all the agents in the cell receive a positive reward. If the action is not correct, all the agents receive a negative reward. The goal is to coordinate the signals sent by the SA to the DA in order to represent the global state of the environment. After a certain number of iterations every agent must know the action that they need to execute in a specific state to obtain positive rewards.

This agent architecture may be used in a diverse set of environments to solve different kind of problems. In our past research work we designed a highly abstract simulation of a distributed sensor network. This environment gave us the opportunity to test the basic feasibility of the agent learning architecture using an abstract environment containing simple network agents. However, the question remained open how the approach would work with more complex and realistic network topologies, traffic patterns and connections. In order to evaluate our learning architecture of agents and to add elements and the complexity of real applications, in this paper we used the network simulator *ns-2* [20], a specifically designed library for ns-2 and the *Tile Coding Software* [22].

To detect the abnormal states that DoS and DDoS generate in a computer network we have slightly modified the original agent architecture as shown in Fig. 1. This architecture is composed by single cell with a Congestion Sensor Agent (CSA), a Delay Sensor Agent (DSA), a Flow Sensor Agent (FSA) and the Decision Agent (DA). We need this diversity of sensor information to develop more reliable IDS. The idea is that each sensor agent perceives different information depending on their capabilities, their operative task and where they are deployed in the network. Furthermore not all the features are available in a single point in the network. Flow and congestion information may be measured in a border router between the Internet and the Intranet whilst delay information may be only available from an internal router. What is more, Flood-Based DDoS attacks are launched from several remote controlled sources trying to exhaust a target's key resource. A stand-alone IDS does not have all the information to accurately identify sources and destinations of DDoS attacks.

In our test domain, the CSA analyses link information on a particular node in the network. It is advisable to use a representative node inside the network topology such as a backbone router or a border router in front of the node or service to protect from intrusions. Specifically this agent samples link utilisation in bytes per second, the size of the queue in packets and the number of packets drop by the queue. These three metrics (link utilisation, queue size and packets drop) are what we call our *feature domain*. To obtain a state representation of the network according with these features we use *tile coding*. The DSA monitors TCP connections between nodes. As previously stated DoS and DDoS attacks modify the normal behaviour of the network in many ways. Some of these changes can be spotted analysing TCP information from connections in the path of the attack. This agent has the same internal structure than the CSA but its feature domain is different. The features analysed for the TCP connections are the average number of ACK packets received, the average window size and the average Round Trip Time (RTT).

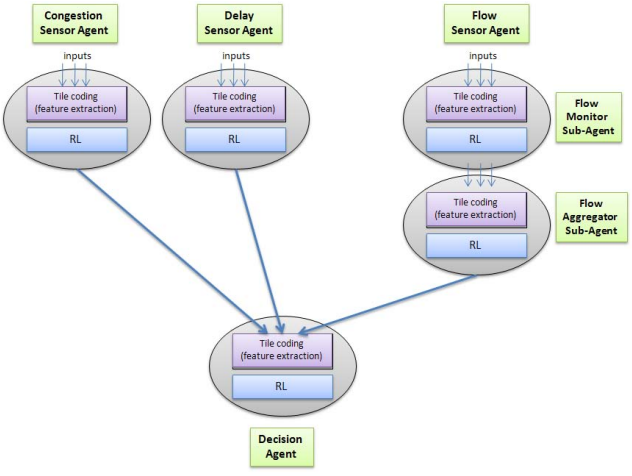

Fig. 1. Agent Architecture

The FSA has a different internal structure than the other sensor agents as can be seen in Fig. 1. This agent is composed by two logical sub-agents, the Flow Monitor (FM) and the Flow Aggregator (FA). The FM analyses the traffic flows that pass through the FSA and its feature domain is composed by protocol number, port number and the average packet size of the flow. Using this information the FM learns which flows are normal traffic and which ones may lead to an attack. The FA aggregates flow information by keeping a flow table with the signals reported by the FM. The feature domain of the FA is very simple. It is the number of attack flows reported by the FM. Finally the original DA described previously does not suffer any modification in its structure, functionality or operation.

4 Tests

To find out whether the agent architecture along with the proposed learning process were capable of detecting abnormal states of the network we performed a series of tests. To add some realistic elements and the complexity of real applications, in this paper we used the network simulator *ns-2* [20]. We generated the network topology of Fig. 2 composed by 7 agents or nodes. Node 0 generates normal FTP-like traffic while node 1 produces normal UDP traffic. Node 4 is an attacker producing a flood of UDP traffic. Node 5 is logically divided in two RL sensor agents, one CSA and one FSA. Their tasks are to forward traffic and collect data about the network. Node 6 is the DA and it solely works as a RL agent. Finally Node 3 is the DSA. It receives valid data from nodes 0 and 1 and it is the node under attack as well.

To measure the success of the performed tests we used a variety of metrics (See Table 1). The most common metrics used to measure the detection performance

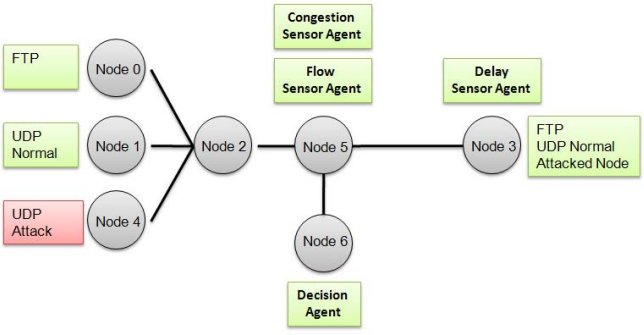

Fig. 2. Tested Network

of IDS are the *false alarm* rate and the *attack detection rate*. The false alarm rate is the fraction of the total alarms that do not represent an intrusion. We will refer to them as *False Positives* (FP) as well. *False Negatives* is the fraction of the total number of intrusions that were not categorized as intrusion. The intrusion detection rate or precision is the fraction of the total number of alarms that were identified as intrusions. To assist us in the design and evaluation of our results we also introduced other prediction metrics commonly used in bioinformatics and machine learning. *Recall* is introduced to show the number of malicious events that the IDS fail to categorise as negative instances. To verify that the IDS is learning how to detect attacks this measure is important to observe. In a similar fashion, accuracy relates all the variables together to an intuitive idea of the performance of the IDS system in relation with the number of correct events categorised. It is important to mention that all the described measures will not properly reflect performance well where the probability of intrusion is very low.

We set up several tests to verify the learning capabilities of the agents as shown in Table 2. We used a control test (Baseline) to train the agents to categorise

Table 1. Performance Metrics 1

Measure	*Formula*	*Meaning*
False Positive Rate	FP / (TP + FP)	The fraction of non negative instances that was redicted as negative
Intrusion Detection Rate	TP / (TP + FP)	The percentage of negative labeled instances that was predicted as negative
Events	TP + TN + FP FN	The total number of events
Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of positive predictions that is correct
Recall	TP / (TP + FN)	The percentage of negative labeled instances that was predicted as negative
Specificity	TN / (TN + FP)	The percentage of predictions that is correct

basic normal and abnormal activity in the network. To simulate the normal traffic we randomly started and stopped connections from node 0 (TCP/FTP) and node 1 (UDP stream). Using another random pattern of connections we used node 4 to simulate the attacks to the network characterised by a flood of UDP traffic. At time $t = 0$ each one of the agents started gathering information from the network and learning as previously explained. At time t_{final} we stopped the learning process and we stored the values of the weight array $\boldsymbol{w}$ in order to use them in each one of the tests of Table 2.

To evaluate the adaptability of the agents we ran test 2 to 8. During these tests the agents are not learning anymore and they are exploiting the knowledge acquired during the training with the baseline test (test 1). Test 2 considers an identical network topology as in test 1 but with different traffic patterns. In this test we modify the start-stop times of the data traffic from the no-attack and attack nodes. Tests 3 to 5 were designed to create a more complex scenario where the attacker changes its attack to mimic authorised or normal traffic. Test 3 simulates when the attacker changes the attack port to any other given port while in test 4 we change the attack port to be the same as the authorised application. In test 5 we simulated when the attacker goes further and changes the attack port and the packet size to mimic the no-attack application. Tests 6 to 8 modify the network topology adding more sources of traffic. These test are important because they modify some of the features that the learning process uses to detect intrusions such as link information, number of flows, packets transmitted per flow type, etc. Specifically in test 6 we added multiple UDP sources and in test 7 we added multiple FTP sources, both of them are valid applications. Finally in test 8 we added multiple attackers as UDP sources to simulate a DDoS attack.

Table 2. Tests

Test	*Description*
1	Baseline test
2	Traffic pattern change
3	Attack port change
4	Attack port same as no attack application
5	Attack port and packet size same as no attack application
6	Multiple valid UDP sources
7	Multiple valid FTP sources
8	Multiple attack sources

5 Results

Unless stated otherwise, we have performed our entire set of tests using the feature domains for sensor and decision agents described in section 2. In Fig. 3 we show the performed tests (1 to 8) evaluated using false positive rate, recall and accuracy. A low false positive rate indicates that our agents will not overwhelm the network operator. A high recall indicates that the agents are able to identify attacks while

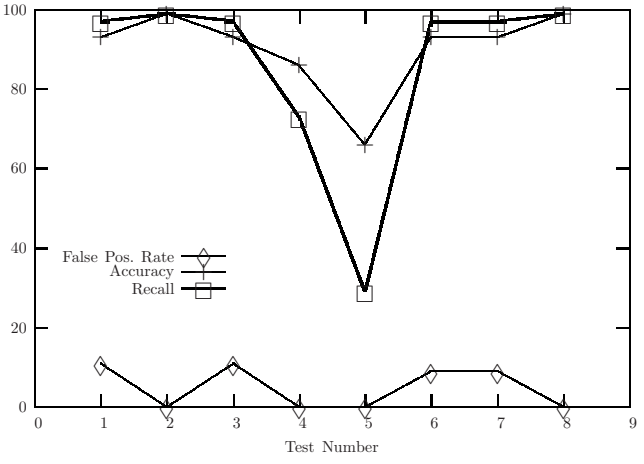

Fig. 3. Test 1-8 and Flow Feature Domain 1

they maintain a low number of false negatives (unidentified attacks). Finally a high level of accuracy indicates that the system is capable of identifying attacks while generating few false positives. The intrusion detection rate (IDR) is another important metric but it can be misleading given a certain type of traffic (e.g. the IDR can be high when the system recognises few attacks but the number of FP is low). Excluding test 5, the remaining tests show acceptable levels of all the intrusion detection metrics including accuracy and recall. Test 2 shows remarkably good levels of accuracy and recall as result of the modified traffic pattern with longer and fewer *no-attack/attack cycles*. A smaller number of no-attack/attack cycles means a small number of FN and FP due to the synchronisation issue between the DA and the collected network information. Contrary to test 2, tests 5 shows a low level of accuracy and recall. Remember that test 5 simulates when the attacker changes the information of the IP packet (protocol, port and packet size) of its attack to mimic a valid connection. In this case when there is an attack the FA interprets the flow information as a no-attack. However the CSA and DSA interpret the network information correctly. When the action signals are transmitted to the DA any of these scenarios may happen:

1. Even though the FA is reporting a no-attack, the signals for the CSA and DSA activate the DA weights that trigger an alarm-action.
2. The signals for the CSA and DSA are not strong enough to activate the alarm-action and the DA triggers a no-alarm-action.
3. The signals from the sensor agents activate weights with similar values for both actions and the DA trigger a *do-not-know action*. In other words, a do-not-know action denotes that the DA does not have enough evidence to trigger a committed action such as an alarm or no-alarm.

In test 5 when the attacks start the congestion and delay value measured by sensor agents are similar to the no-attack states causing the DA to trigger an

incorrect action generating a FN. As the attack progresses the congestion and delay information make the current state appear to the DA as a no-attack but not strong enough to trigger a no-alarm action. Instead, the DA triggers the do-not-know action. Finally when the attack is at its peak, the signals from the DSA and CSA make the value of the alarm action better than the no-alarm and the DA triggers the alarm. A similar behaviour takes places when the attacker slows down its attack.

Trying to improve the intrusion detection metrics we ran more tests changing the feature domain of the sensor agents. This task showed the difficulty of choosing an optimal set of features, as in many applications of machine learning. While some sets improved the metric for some tests, they also showed worse results for other tests. The set of features presented in the past section yielded the best results overall.

In order to compare our learning IDS to alternative approaches, we implemented two common hand-coded (i.e., non-adaptive) IDS techniques. The first hand-coded approach (Hand-Coded 1) emulated a misuse IDS. In this case the IDS is looking for the patterns that match an attack in the same way that some commercial misuse IDS do in real world networks, e.g. Snort [21] and Checkpoint [6]. To evaluate a more complex IDS implementation the Hand-Coded 2 approach integrates the same variety of input information as our learning algorithm. This approach is similar to the one employed in some commercial Intrusion Protection System (IPS) such as the Cisco Intrusion Prevenstion System Sensor [7]. These type of devices search for intrusions through signature and anomaly detection methods. We evaluated the learning and hard-coded approaches using test 2 and test 5. We used test 2 because it only changes the traffic pattern of the attack and it must be very simple to detect. Attacks in test 5 are the hardest to detect because they emulate some of the signatures of normal traffic. The learning curves of the test are shown in Fig.4.

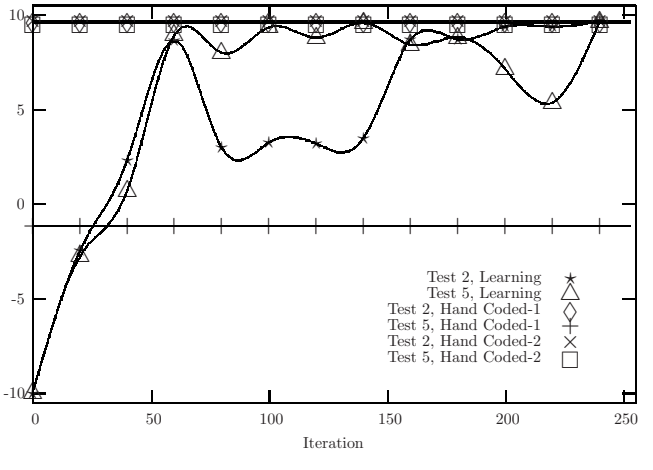

Fig. 4. Learning Curves

The Hand-Coded 1 approach had no problem to identify attacks and had low false negatives for test 2 but it completely failed to detect attacks in test 5. This is the same problem that misuse IDS have when the signature of the attack changes or when they face unknown attacks. The results for Hand-Coded 2 and our learning approach confirm our argument that for more reliable intrusion detection we need a variety of information sources. On test 5 the learning and the Hand Coded 2 approaches were capable of detecting the attacks even though one of the sensors was reporting incorrect information. This scenario also could be seen as the emulation of a broken or comprimesed sensor forced to send misleading signals.

Both the Hand-coded 2 and learning approaches present very good results regarding the identification of normal and abnormal states in the network. Hand-coded 2 reaches maximum performance from the beginning of the simulation. Nevertheless it has a major drawback, it requires in-depth knowledge from the policy programmer about the the network traffic and patterns in order to detect intrusions. While the learning algorithm requires some time to learn to recognise normal and abnormal activity, it does not require any previous knowledge about the behaviour of the network or exactly which features to observe. Although different sets of features show different results, the learning approach's flexibility allows the use of any (large enough) set of features to achieve some reasonable level of detection. The learning approach automatically will use the interesting features to detect attacks and it will ignore the ones that do not represent different states.

6 Related Work

Problems such as the curse of dimensionality; partial observability and scalability in MARL have been analysed using a variety of methods and techniques [15,18] and they represent the foundation of our research. More recent work related with our research include the use of Hierarchical Reinforcement Learning [10], learning automata [13] and game theory [16]. An application of MARL to networking environments is presented in [3] where cooperative agents learn how to route packets using optimal paths. Using the same approach of flow control and feedback from the environment, other researchers have expanded the use of RL to explore its use to control congestion in networks [8,11], routing using QoS [9] and more recently to control DDoS attacks [25].

The use of RL in the intrusion detection field has not been widely studied and even less in distributed intrusion detection. On IDS with RL research we found [4,5] where the authors trained a neural network using RL and aplied CMAC as the function approximation technique and [1] where game theory is used to train agents to recognise DoS attacks against routing infrastructure. Other recent research work include the use of RL to detect host intrusion using sequence system calls [26] and the previously mentioned [25].

7 Conclusion and Future Work

In this paper we have shown how a group of agents can coordinate their actions to reach the common goal of network intrusion detection. During this process decision agents learn how to interpret the action-signals sent by sensor agents without any previously assigned semantics. These action-signals aggregate the partial information received by sensor agents and they are used by the decision agents to reconstruct the global state of the environment. In our case study, we evaluate our learning approach by identifying normal and abnormal states of a realistic network subjected to various DoS attacks. Overall the following conclusions can be drawn:

- We have successfully applied RL in a group of network agents under conditions of partial observability, restricted communication and global rewards in a realistic network simulation.
- The use of a variety of network data has generated good results to identify the state of the network. The system presents high reliability even in cases when some sensor information is missing or compromised.
- The learning approach yields better results than the simple hand coded alternative. It also yields similar results to a more complex hand coded alternative using a variety of sensor information. The main advantage of the learning approach is that it does not need a trainer with prior knowledge of the network environment.

Future work includes evaluating on-line learning (i.e. during deployment of the IDS) and scaling up our learning approach to a large number of agents using the hierarchical approach from our previous work on abstract networks. This will allow us to create more complex network topologies emulating geographical cells of agents, security domains composed by cells or groups of cells, complex DDoS attacks and eventually the emulation of real packet streams inside the network environment.

References

1. Awerbuch, B., Holmer, D., Rubens, H.: Provably Secure Competitive Routing against Proactive Byzantine Adversaries via Reinforcement Learning. John Hopkins University, Tech. Rep. (May 2003)
2. Barford, P., Jha, S., Yegneswaran, V.: Fusion and filtering in distributed intrusion detection systems. In: Proceedings of the 42nd Annual Allerton Conference on Communication, Control and Computing (September 2004)
3. Boyan, J., Littman, M.: Packet routing in dynamically changing networks: A reinforcement learning approach. Advances in Neural Information Processing Systems 6, 671–678 (1994)
4. Cannady, J.: Applying CMAC-based on-line learning to intrusion detection. In: Proceedings of the International Joint Conference on Neural Networks, vol. 5, pp. 405–410 (2000)
5. Cannady, J.: Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks. In: Proc. 23rd National Information Systems Security Conference (2000)

6. CheckPoint. CheckPoint, N.G.X.: Firewall SmartDefense (June 2008), http://www.checkpoint.com/products/ips-1/index.html
7. Cisco. Configuring Anomaly Detections (June 2008), http://www.cisco.com/en/US/docs/security/ips/6.1/configuration/guide/cli/cli_anomaly_detection.html
8. Dowling, J., Curran, E., Cunningham, R., Cahill, V.: Using feedback in collaborative reinforcement learning to adaptively optimize MANET routing. Systems, Man and Cybernetics, Part A, IEEE Transactions on 35(3), 360–372 (2005)
9. Gelenbe, E., Lent, M., Su, R.: Autonomous smart routing for network QoS. In: Proceedings of International Conference on Autonomic Computing 2004, pp. 232–239 (2004)
10. Ghavamzadeh, M., Mahadevan, S., Makar, R.: Hierarchical multi-agent reinforcement learning. Autonomous Agents and Multi-Agent Systems 13(2), 197–229 (2006)
11. Hwang, K., Tan, S., Hsiao, M., Wu, C.: Cooperative Multiagent Congestion Control for High-Speed Networks. Systems, Man and Cybernetics, Part B, IEEE Transactions on 35(2), 255–268 (2005)
12. Institute, S.: Sans top-20 2007 security risks, 2007 annual update (2008)
13. Katja Verbeeck1, P.V., Nowe, A.: Networks of learning automata and limiting games. In: Adaptive Learning Agents and Multi Agent Systems 2007, pp. 171–182 (2007)
14. Mirkovic, J., Reiher, P.: D WARD, A Source-End Defense against Flooding Denial of Service Attacks. Dependable and Secure Computing, IEEE Transactions on 2(3), 216–232 (2005)
15. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems 11(3), 387–434 (2005)
16. Powers, R., Shoham, Y.: New criteria and a new algorithm for learning in multi-agent systems. Advances in Neural Information Processing Systems 17, 1089–1096 (2005)
17. Servin, A.L., Kudenko, D.: Multi-agent Reinforcement Learning for Intrusion Detection. In: Tuyls, K., Nowe, A., Guessoum, Z., Kudenko, D. (eds.) ALAMAS 2005, ALAMAS 2006, and ALAMAS 2007. LNCS (LNAI), vol. 4865, pp. 211–223. Springer, Heidelberg (2008)
18. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artificial Intelligence 171(7), 365–377 (2007)
19. Siaterlis, C., Maglaris, B.: Towards multisensor data fusion for dos detection. In: Proc. of the 19th ACM Symposium on Applied Computing, Nicosia, Cyprus, pp. 439–446 (2004)
20. N. Simulator. 2 (NS2) (January 2008), http://www.isi.edu/nsnam/
21. I. SourceFire. Snort (June 2008), http://www.snort.org/
22. Sutton, R.: Tile Coding Software, Version 2.0 (2007)
23. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
24. Watkins, C., Dayan, P.: Q-learning. Machine Learning 8(3), 279–292 (1992)
25. Xu, X., Sun, Y., Huang, Z.: Defending DDoS Attacks Using Hidden Markov Models and Cooperative Reinforcement Learning. In: Yang, C.C., Zeng, D., Chau, M., Chang, K., Yang, Q., Cheng, X., Wang, J., Wang, F.-Y., Chen, H. (eds.) PAISI 2007. LNCS, vol. 4430, p. 196. Springer, Heidelberg (2007)
26. Xu, X., Xie, T.: A Reinforcement Learning Approach for Host-Based Intrusion Detection Using Sequences of System Calls. In: Proceedings of the International Conference on Intelligent Computing (2005)