Effective multimodel anomaly detection using cooperative negotiation

Alberto Volpatto, Federico Maggi, and Stefano Zanero

Dipartimento di Elettronica e Informazione
Politecnico di Milano
{volpatto,fmaggi,zanero}@elet.polimi.it

Abstract. Many computer protection tools incorporate learning techniques that build mathematical models to capture the characteristics of system's activity and then check whether live system's activity fits the learned models. This approach, referred to as *anomaly detection*, has enjoyed immense popularity because of its effectiveness at recognizing unknown attacks (under the assumption that attacks cause glitches in the protected system). Typically, instead of building a single complex model, smaller, partial models are constructed, each capturing different features of the monitored activity. Such *multimodel* paradigm raises the non-trivial issue of combining each partial model to decide whether or not the activity contains signs of attacks. Various mechanisms can be chosen, ranging from a simple weighted average to Bayesian networks, or more sophisticated strategies. In this paper we show how different aggregation functions can influence the detection accuracy. To mitigate these issues we propose a radically different approach: rather than treating the aggregation as a calculation, we formulate it as a decision problem, implemented through cooperative negotiation between autonomous agents. We validated the approach on a publicly available, realistic dataset, and show that it enhances the detection accuracy with respect to a system that uses elementary aggregation mechanisms.

Keywords: Anomaly detection, cooperative negotiation.

1 Introduction

As a growing number of business and personal activities are conducted over the Internet, cybercrime has became a growing concern [1]. In the current threat landscape, 0-day exploits and site-specific attacks are at the same time the most challenging and frequent threats [2]. 0-day exploits take advantage of vulnerabilities not publicly disclosed, whereas site-specific attacks target a specific application, such as custom product, rather than a widely deployed system. Among the defense tools, anomaly detectors leverage a description of the normal activity of the protected system, and are therefore potentially effective against attacks never seen before. The assumption (and, to some extent, the limitation) is that attacks would leave traces that can be automatically recognized as anomalous.

On the other hand, misuse-based detectors, which rely on a list of signatures of *known* attacks, only offer protection against attack vectors that have been publicly disclosed. Despite its remarkable precision at detecting known threats, the misuse-based paradigm offer no help against 0-day and custom-made attacks.

A long-standing approach in the design of complex systems is the use of multiple models [3, 4]. This paradigm is also effective for anomaly detection, as proven by the recent literature detection [5–12] and lies in the decomposition of a complex problem (i.e., capturing the "normal behavior" of a potentially large and complex system) into simpler sub-problems, each tackled by one mathematical model. For example, there are systems that analyze the observed activity (e.g., the HTTP requests and responses) and build models such as the average length of a request, or the relative frequency of ASCII symbols into an HTTP request body. Usually, such models have a numerical representation that contributes to an overall evaluation of the degree of anomaly. The models can be encapsulated into autonomous agents [13]. This technique have been shown to be as effective as the traditional, multimodel approach [14–19], with the benefit of a more natural design and the existence of a plethora of multiagent programming frameworks [20]. Unfortunately, while the use of multiagent systems may improve the ease of design of anomaly detectors, the problem of aggregating the models persists.

The core point of this paper is that the aggregation phase is crucial to achieve good precision. We propose to tackle this problem and its issues (described in Section 2.1) by exploiting the multiagent framework beyond its architectural properties. More precisely, we use cooperative negotiation to implement the aggregation task as a decision problem. This approach is inspired by the seminal study described in [21], where cooperative negotiation has been used to classify the payloads of TCP packets. However, the study was explicitly a toy example, tested on artificial, outdated traffic (i.e., IDEVAL [22]). We further develop the idea, adapt it to real-world settings, and examine carefully the impact of the negotiation protocol's parameters. The contributions of this paper can be summarized as follows.

– In Section 2 we discuss the issues caused by simple aggregation strategies, and motivate why this problem should be addressed to achieve reliable attack detection.
– In Section 3 we present a simple, yet very effective, technique that leverage cooperative negotiation between autonomous agents to decide whether or not a given event is an attack and detail how we incorporated this technique in an anomaly detector.
– In Section 4 we evaluate the detection capabilities of the tool obtained over a realistic, publicly-available data set, and discuss its limitations.

The results of our experiments show that the proposed technique alleviates the detection errors caused by inaccurate combinations of models. Our approach applies to any multimodel system. However, due to the popularity of web applications and web-based attacks, we validate it on a recent anomaly detector designed to protect web applications.

2 Multimodel anomaly detection

A learning-based anomaly detector learns the normal activity by observing a system's activity. In the representative, simple example of HTTP, such activity consists in the HTTP requests and responses exchanged between servers and clients. Requests, or *queries*, $Q = \{q_1, q_2, \ldots, q_j, \ldots\}$, are usually decomposed into resources (i.e. paths) and parameters. For instance, the request `'GET /page?uid=u44&p=14&do=delete'` contains the resource `'/page'` and the parameters $\{\langle \texttt{uid}, \texttt{'u44'}\rangle, \langle \texttt{g}, 14\rangle, \langle \texttt{do}, \texttt{'delete'}\rangle\}$.

During an initial *learning* (or training) phase, a multimodel detector instantiates different mathematical models, $m^{(1)}, \ldots, m^{(Z)}$, to compute certain features, $1, \ldots, Z$, on each training sample (e.g., an HTTP request, response, or a sequence of requests-responses). The specific models and the strategy to combine their output determine the classes of attacks that can be detected. Typical models proposed in literature capture features such as the average length of the string parameters, their character distribution and probabilistic grammar, or the order in which parameters appear across the requests against the each resource. Unfortunately, due to space limitations, we must refer the interested reader to [9,12,23] for more details. During *detection*, the model instances computed during training are used as maps, $m^{(z)} : Q \mapsto [0, 1], \forall z$, and their outputs are aggregated into an overall *anomaly score*. This score is checked against thresholds and alerts are fired accordingly. Thresholds are fixed a priori or, usually, computed during learning. The improvements proposed in this work — validated using the set of models implemented in Masibty [12] — are independent from the particular models, and thus can be easily applied to any learning-based detector.

2.1 Drawbacks of model aggregation

In the design of a multimodel anomaly detector, the value aggregation [24] phase has a significant impact on the quality of detection results. Just for simplicity in illustrating our point, we can roughly distinguish between simple and complex aggregation strategies, and in both cases such strategies can be parametric or non-parametric.

An example of simple, non-parametric aggregation strategy is the arithmetic mean. While this type of approach requires no user intervention and is very simple to understand, the lack of a differentiation between the models does not allow to control the impact of models with poor performance on the overall detection. The most natural solution to this issue is the use of parametric aggregation functions (e.g., a weighted average), with different parameters assigned to the models in order to optimize detection quality. These mechanisms obviously create the non-trivial problem of choosing the parameters (e.g., the weights), which clearly influence the final value. In addition, if the system employs a large set of models, it might be difficult to properly set all the weights. Although the optimal weights can be computed automatically from the data, as demonstrated by the results of the experiment described in Section 4.1, this is not sufficient to achieve good detection accuracy if a simplistic aggregation method is used. More complex strategies, such as Bayesian networks (or ad-hoc aggregation criteria), can perform very well under the conditions they have been designed for, but unfortunately their inherent complexity makes manual tuning and improvements difficult for the end-users. Our aggregation approach, which performances are comparable to such methods, have the advantage of being easier to manually configure.

From the previous observations, it follows that the design of a simpler, generic method to reduce detection errors caused by model aggregation is necessary to obtain a reasonable level of protection from multimodel anomaly detectors.

3 Exploiting cooperative negotiation

We implemented our model aggregation approach by modifying the training and detection algorithms of Masibty [12].

3.1 Modifications to the learning phase

To fully implement the detection phase detailed in Section 3.2, the learning phase of a traditional multimodel detector requires some minor modifications. In particular, every model, $m^{(z)}$, must implement a *trust model*, $T_j^{(z)} : Q \mapsto [0,1]$, that assesses the "reliability" of $m^{(z)}$, up to the j-th training step (i.e., after the j-th training has been analyzed). As motivated in Section 3.2, this model is required for completing the cooperative negotiation correctly. The final decision indeed depends upon the agents with higher trust level, while the impact of poorly-trained agents (which may lead to overfitting) is minimized. For this reason, the trust functions must be designed in such a way that models that have received ample training are assigned high trusts, since they are likely to produce accurate and reliable detections. For example, one of the models checks for the presence of parameters in each HTTP request and computes their appearance ratio across all the requests. The anomaly score is calculated as $1 - \min\left(\frac{M}{R}, \min\left(\frac{P}{R}\right)\right)$, where M and P, respectively, indicate the number of missing and present parameters, while R is the total number of requests. The trust level is high if the presence is nearly constant, while it decreases if an application exhibits variations. The trust function is thus $1 - \frac{M}{R}$. Due to space limitations, we refer the reader to [12] for details on the models implemented on the prototype (see Fig. 1 in [12]) used for our experiments.

Optionally, the trust level can also be exploited to optimize training, by stopping it automatically when a sufficient amount of data is received by each model. For example, we adopted $\delta_W(j) := \max_{j \in W} T_j^{(z)} - \min_{j \in W} T_j^{(z)}$, where W is a sliding window[1]. By choosing a small $\varepsilon > 0$ (we used $\varepsilon = 0.003$ and $W = 5$), a model is considered stable after the j-th training sample if $\delta_W(j) \leq \varepsilon$. Although more sophisticated criteria can be designed, we noticed that, under

[1] the sliding window size influences the training duration: smaller values tend to stop training early, while higher values result in a longer and more conservative training

the conditions described in Section 4, this is sufficient to achieve good detection results. This optimization, however, is not necessary to adopt our cooperative negotiation approach, and we will evaluate its impact separately.

3.2 Modifications of the detection phase

We translate the value aggregation problem that arises during detection into a decision problem implemented via *cooperative negotiation* between autonomous agents. In artificial intelligence, an *agent* is the abstraction of an entity capable of reading inputs by observing the environment, and to perform actions toward the achievement of certain goals. In our context, the *environment* is the network segment which the agent receives HTTP messages from. No other input than the HTTP messages is passed to the agents. The *goal* is to find the correct degree of anomaly, that is, the numeric value that would minimize the detection errors. *Multiagent* systems comprise a *coordination* protocol, implemented by the agents (and a mediator, if present) to achieve a *global* goal. In general, these protocols can be *competitive*, when the goals of the agents are conflicting, or *cooperative*, when the agents pursue a common goal. A form of coordination is *negotiation*, where the agents tend to "harmonize" conflicting goals toward the achievement of the global goal.

We specifically propose to use a *cooperative negotiation protocol*, described in the following, to reach an agreement on the anomaly score used to classify HTTP messages as benign or malicious.

Cooperative negotiation protocol Our system comprises n agents, $M_1, \dots,$ $M_i, \dots, M_n$, and a mediator M. Each agent embeds exactly one of the partial models implemented in the original prototype [12, Section IV], and communicates only with the mediator (which embeds no models).

The protocol is initiated every time a new sample, q_j, is observed. The j-th session iterates multiple times. Each iteration is denoted with t and begins at $t = 0$. We define the partial degree of anomaly $p_i^t \in [0, 1]$ as the degree of anomaly computed by the i-th agent using its embedded model $m^{(z_i)}$ at iteration t. The protocol proceeds as follows:

1. each M_i receive the sample q_j and calculates its offer, $p_i^t = m^{(z_i)}(q_j)$.
2. p_i^t is sent to M along with the agent's trust level, $w_i = T^{(z_i)}$.
3. M receives p_i^t and w_i, $\forall i = 1, \dots, n$, and calculates an agreement a^t by using the agreement function $a^t = A(p_1^t, w_1, p_2^t, w_2, \dots, p_n^t, w_n)$.
4. M sends its counter offer a^t to all the agents.
5. Each M_i calculates the new offer p_i^{t+1} by using a negotiation function $p_i^{t+1} = F_i(p_i^t, a^t)$.
6. Each M_i sends p_i^{t+1} to M, and steps 3. to 6. are repeated until an agreement, a, for the j-th session, $a = a_j$, is reached (i.e., until $p_i = p_{i'}, \forall i \neq i'$).

The global evaluation of the anomaly score for the sample q_j is thus a_j.

We define the *agreement function* to be the weighted average of the offers, where weights are the trust levels: $A(p_1^t,\, w_1,\, p_2^t,\, w_2,\, \ldots,\, p_n^t,\, w_n) := \frac{\sum_{i=1}^n p_i^t w_i}{\sum_{i=1}^n w_i}$. The trust level is *constant* throughout the negotiation.

The *negotiation function* is $F_i(p_i^t, a^t) = p_i^t + \alpha_i(a^t - p_i^t)$, where $a^t - p_i^t$ expresses a measure of disagreement between agent M_i and the global system. Note that p_i^{t+1} is determined only by values at time t: for this reason, and because each agent does not change the evaluation of its embedded model over time (for a given q_j), it is unnecessary to actually perform a two-way communication between the agents and the mediator. Instead, upon receiving all the initial offers, the mediator can run the negotiation protocol without communicating with the agents. The cooperative negotiation protocol described does not impose any negotiation strategy to the agents, that is indeed related to the implementation of both functions detailed above.

The *agreement coefficient*, $\alpha_i \in (0, 1)$, expresses the willingness of agent M_i to propose its offer versus the counter-offer received from the mediator. When $\alpha_i \to 0$ each agent tends not to modify its offer, while $\alpha_i \to 1$ causes each agent to agree with the counter-offer. We compute the agreement coefficient as a function of the trust level of the agent's embedded model. If the trust level is close to 0, then its evaluation would not be reliable; thus, during the negotiation session its influence should be minimized to "ignore" its offers. On the other hand, a trust level close to 1 means maximum reliability. A sigmoid-shaped function $f_\alpha(w_i) = \frac{1}{1+e^{h(w_i-k)}}$ is a good implementation of the above rationale, where h is the smoothness and k is the central value.

It can be shown that h has negligible impact on the final agreement (in our experiments, we used $h = 7.5$) and only influences the speed of the negotiations. We adopted $k = 0.5$ to express that all the agents have neutral impact on the agreement against the mediator, and such impact only depends on the trust levels computed from data during training. In other words, 0.5 is the natural value of k if one needs to avoid biased detections. It can be shown that also k has a negligible impact on the results. We demonstrate both observations with an experiment described in Section 4.2.

The cooperative negotiation mechanism described is proved to be *connectively stable* in [25]. This means that agents will reach a stable agreement on attack probability, regardless of their initial offer and for any n. In order to prove our point that just by modifying the negotiation protocol we could improve the quality of the results, we did not modify the original detection phase of the system any further. In particular, we have not modified the way global anomaly thresholds are calculated and used to raise alerts.

In addition to the traditional sensitivity parameter, used in every anomaly-based detector to trade off *False Positive Rate* (FPR) versus *Detection Rate* (DR), h and k are the only parameters strictly required by our approach, and their choice has a very limited impact on the system performance. W and ε are required only for optimized learning.

4 Evaluation

To validate our approach, we conducted two experiments on the traffic[2] captured during the International Capture the Flag 2008, organized by the University of California, Santa Barbara. The traffic, mostly HTTP, contains 0-day vulnerabilities, and the majority of the players were skilled hackers, able to prepare custom and diverse exploits. Unfortunately, this causes the lack of any ground truth other than the list of (known) attacks detectable with the Snort misuse-based system. To alleviate this issue we used the portion of the dataset that contains clean background traffic, and injected custom, real-world attacks during detection. In this way, the ground truth is perfectly known. The background traffic contains 44,102 HTTP messages, i.e., 22,051 request-response couples. We cross-validated our system by using 14,961 request-response couples for training and 7,090 for detection. More precisely, we injected instances of the three most common types of attacks against web applications[3], such as cross-site scripting (XSS) (e.g., CVE-2009-0781), SQL injections (e.g., CVE-2009-1224), and command injections (e.g., CVE-2009-0258). The XSS attacks are variations on those listed in [26], the SQL injections were created similarly from [27], and the command execution exploits are variations of common command injections against the Linux and Windows platforms. In addition to cross-validation, to avoid biased experiments on the same attack instances, the injected strings are randomly drawn from a set of alternatives. In particular, we used 14 different SQL injection vectors, 4 command injections, and 94 XSSs. Note that, this is similar to the use of variants of the same dataset, as it avoids using the same exact set of attacks over and over. To this end, using a uniform probability distribution, we randomized (1) the type of attack, (2) the HTTP parameter to use for the injection, and (3) the vector (of a given type) to inject. The resulting traffic contains 1,000 randomized attacks in every experiment.

4.1 Benefit of cooperative negotiation

This experiment aims at showing that our approach can effectively mitigate detection errors better than a technique based solely on value aggregation (i.e., weighted average). To this end, we ran the original prototype in its original configuration, as described in [12], then with the modifications described in Section 3, and finally implementing also the optimization for the learning phase briefly described in Section 3.1 (which is optional).

The comparison is show in Fig. 1, with a ROC curve showing DR and FPR for different working points. As it can be seen, the cooperative negotiation dramatically improves the classification accuracy with respect to the simplistic weighted average. For instance, at $FPR = 0.1$, the cooperative negotiation yields an increment of about 54% on the DR. It must be noted that, in the paper that

[2] available for download at `http://ictf.cs.ucsb.edu/data/ictf2008/`

[3] `http://owasptop10.googlecode.com/files/OWASPTop10-2010.pdf`

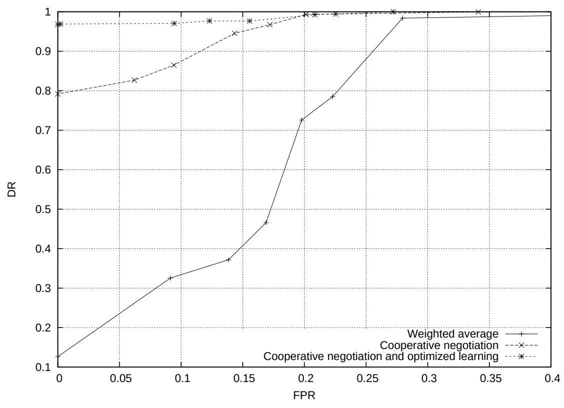


Fig. 1. ROC curve of the original prototype (solid line) and with the modifications we propose (dashed line), and with optimized training (dotted line).

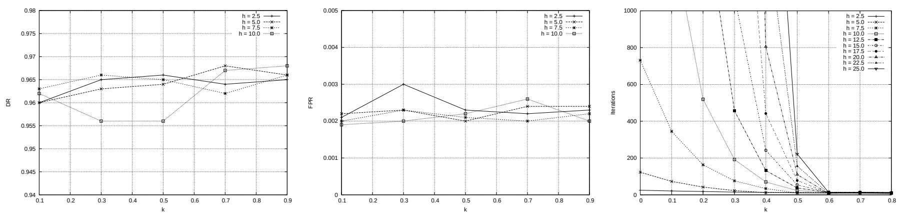
describes the original prototype, the experiments leveraged a completely artificial, small dataset, comprising only a limited number of attacks: this explains why the original system performs significantly worse in this experiment.

The accuracy can be further improved by adopting the optimization of the learning phase (see Section 3.1), although this is not required to apply the cooperative negotiation.

4.2 Influence of the parameters

This experiment shows that the parameters introduced by our approach, i.e., the smoothness, h, and the central value, k, of the alpha function, f_α, only marginally impact the detection quality. In addition, we provide guidance to choose these parameters to minimize the negotiation overhead. To this end, we used all of the modifications described in Section 3 (with optimized learning) at a fixed ROC working point, and varied $h \in \{2.5, 5, 7.5, 10\}$, $k \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. As shown in Fig. 2, the influence of these parameters on (a) DR and (b) FPR is barely noticeable.

Rather than providing a theoretical complexity boundary to the computational overhead induced by the negotiation process, we estimate it under real conditions. As shown in Fig. 2(c), h and k have significant impact on the computational overhead (e.g., number of iterations I), necessary to complete the negotiation. However, as seen in Fig. 2(a-b), these parameters have almost *no* impact on the detection accuracy. Thus, setting $k \geq 0.5$ and choosing a safe value of h allows to limit the number of iterations. For example, in the experiment discussed in Section 4.1 we used $h = 7.5$.



(a) $DR = DR_h(k)$ (b) $FPR = FPR_h(k)$ (c) $I = I_h(k)$

Fig. 2. Negligible influence of k and h on the detection quality of our system (a-b) and noticeable influence on the length of the negotiation, I.

4.3 Discussion and limitations

It is important to clarify the main limitations of our approach. Firstly, an incorrect choice of h and k may cause the negotiation protocol not to terminate in a reasonable amount of time. However, in Section 4.2, we show that these parameters do not influence the detection quality and, more importantly, provide a guidance for setting h and k to safe values. Secondly, as described in Section 3.2, the trust level is constant throughout all the negotiations. In our opinion, this does not impact the detection accuracy significantly, although this needs to be assessed thoroughly.

From our experiments we conclude, however, that the technique described in Section 3 effectively improves the detection capabilities of a multimodel anomaly detector.

5 Related work

Anomaly-based approaches have been proposed to protect computer systems from attacks by exploiting learning algorithms in different veins. Ensembles of simple models (e.g., character distribution) are effective at capturing the normal characteristics of computer programs [10, 11], network traffic [7], or HTTP messages [9, 12], respectively. Unfortunately, due to space limitations, we must refer the reader to a comprehensive survey on anomaly detection [28].

Multiagent systems are very useful for implementing complex systems [14] and have been applied also to intrusion detection as a handy replacement of classic multimodel architectures. For instance, in [13] an agent is assigned to each system component or task (e.g., network sniffing, stream reassembly). However, in this type of approaches, only the architecture of a multiagent system is exploited. Instead, our approach exploits multiagent systems as a *paradigm*: not only each agent embeds a detection procedure, but proper algorithms drawn from artificial intelligence are leveraged to perform the decision. One of the first attempts of translating the intrusion detection problem to an interaction between agents appeared in [15], where the use different classes of agents, which do not communicate to each other, is used to detect different types of anomalies, and to aggregate the decisions through a special agent. Unfortunately, some malicious activity (e.g., distributed attacks or evasion attacks) are difficult to detect if the agents do not communicate, since each agent has a limited view of the attacks. This idea is improved in [16, 18] by introducing communication between agents and by embedding a Bayesian network into each agent. Different decision techniques have been embedded in the multiagent detector described in [17], where special agents called "decisors" use fuzzy inference to bid for the most appropriate actions to counteract the anomalies reported by other agents. CAMNEP [19] has two important analogies with our work, since each agent (1) learns a different model (similar to those cited in Section 2) and, (2) is assigned a trust level that reflects the completeness of its training. Unfortunately, this approach do not fully exploit artificial intelligence algorithms as results from each agents are simply averaged. The exploratory work described in [21] uses the intrusion detection task as a case study to apply cooperative negotiation algorithms to detect attacks. Although the results are promising, the approach reduces the computation of the trust to a constant function. More importantly, the agreement coefficient devised by the authors causes the agents to agree on results that once again tend to approximate a weighted average. Recent proposals focused on updating the learned specifications dynamically at run-time, hence requiring no or little human intervention also in the case of concept drifts [29, 30]. Among this research line, the technique described in [31] is applied to the aforementioned CAMNEP multiagent anomaly detector. However, although these model-updating techniques can certainly improve the accuracy of an existing detector such as the one described in this paper, they do not constitute a new *detection* mechanism *per sé*, while in this work we focused on multiagent algorithms for designing effective detection strategies.

6 Conclusions

In this work, we have analyzed an issue that occurs in virtually any multimodel, anomaly-based intrusion detection system: the detection errors caused when the outputs of each partial model are aggregated together to form a global evaluation, used to decide whether or not a certain event is an attack. We proposed to embed detection models into separate agents, in a multiagent system, and then to exploit cooperative negotiation to implement a more robust value-aggregation strategy. To test our approach, we modified a web anomaly detection system which used a simple weighted average, to use cooperative negotiation.

The results obtained by testing the original versus the modified system are promising. More precisely, our approach can improve the detection rate dramatically at parity of false positive rate. In addition, the detection quality is not influenced by new parameters we introduced, thus our approach does not require any further tuning effort and could be effortlessly applied to any learning-based anomaly detector that employs simpler aggregation approaches.

As future work, besides addressing the limitations discussed in Section 4.3, we plan to evaluate the performance overhead introduced by our approach more thoroughly, with particular attention to the time necessary to complete the negotiation phase and the comparison to other simpler, yet less formal, aggregation methods.

Acknowledgments

The authors are thankful to N. Basilico and F. Amigoni. This work has been partially supported by the European Commission through IST-216026-WOMBAT funded by the 7th FP. The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

References

1. Carr, J.: Inside Cyber Warfare: Mapping the Cyber Underworld. O'Reilly Media, Inc. (2009)
2. The SANS Institute: Zero-day vulnerability trends. `http://www.sans.org/top-cyber-security-risks/zero-day.php` (September 2009)
3. Fishwick, P.A.: An integrated approach to system modeling using a synthesis of artificial intelligence, software engineering and simulation methodologies. ACM Trans. Model. Comput. Simul. **2**(4) (1992) 307–330
4. Fishwick, P.A., Zeigler, B.P.: A multimodel methodology for qualitative model engineering. ACM Trans. Model. Comput. Simul. **2**(1) (1992) 52–81
5. Denning, D.E.: An Intrusion-Detection Model. IEEE Transactions on Software Engineering **13**(2) (1987) 222–232
6. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security **3**(4) (2000) 227–261
7. Kruegel, C., Toth, T., Kirda, E.: Service-Specific Anomaly Detection for Network Intrusion Detection. In: Proceedings of the Symposium on Applied Computing (SAC 2002), Spain (March 2002)
8. Kruegel, C., Mutz, D., Valeur, F., Vigna, G.: On the detection of anomalous system call arguments. In: Proceedings of the 2003 European Symp. on Research in Computer Security, Gjøvik, Norway (Oct. 2003)
9. Kruegel, C., Robertson, W., Vigna, G.: A Multi-model Approach to the Detection of Web-based Attacks. Journal of Computer Networks **48**(5) (July 2005) 717–738
10. Mutz, D., Valeur, F., Kruegel, C., Vigna, G.: Anomalous System Call Detection. ACM Transactions on Information and System Security **9**(1) (February 2006) 61–93
11. Maggi, F., Matteucci, M., Zanero, S.: Detecting intrusions through system call sequence and argument analysis. IEEE Transactions on Dependable and Secure Computing **99**(PrePrints) (2008)
12. Criscione, C., Maggi, F., Salvaneschi, G., Zanero, S.: Integrated detection of attacks against browsers, web applications and databases. In: European Conference on Computer Network Defence - EC2ND 2009. (2009)
13. Helmer, G., Wong, J.S.K., Honavar, V.G., Miller, L., Wang, Y.: Lightweight agents for intrusion detection. J. Syst. Softw. **67**(2) (2003) 109–122
14. Jennings, N.R.: An agent-based approach for building complex software systems. Commun. ACM **44**(4) (2001) 35–41

15. Spafford, E., Zamboni, D.: Intrusion detection using autonomous agents. Computer Networks **34**(4) (2000) 547–570
16. Ghosh, A., Sen, S.: Agent-based distributed intrusion alert system. In: Distributed Computing - IWDC 2004. Volume 3326/2005 of Lecture Notes in Computer Science., Heidelberg, Springer Berlin (2004) 240–251
17. Dasgupta, D., Gonzalez, F., Yallapu, K., Gomez, J., Yarramsettii, R.: CIDS: An agent-based intrusion detection system. Computers & Security **24**(5) (2005) 387–398
18. Gowadia, V., Farkas, C., Valtorta, M.: PAID: A probabilistic agent-based intrusion detection system. Computers & Security **24**(7) (2005) 529–545
19. Rehak, M., Pechoucek, M., Celeda, P., Novotny, J., Minarik, P.: Camnep: agent-based network intrusion detection system. In: AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2008) 133–136
20. Allan, R.J.: Survey of agent based modelling and simulation tools. Technical report, STFC Daresbury Laboratory, Daresbury, Warrington WA4 4AD (May 2010)
21. Amigoni, F., Basilico, F., Basilico, N., Zanero, S.: Integrating partial models of network normality via cooperative negotiation: An approach to development of multiagent intrusion detection systems. In: WI-IAT '08, Washington, DC, USA, IEEE Computer Society (2008) 531–537
22. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Comput. Networks **34**(4) (2000) 579–595
23. Song, Y., Stolfo, S., Keromytis, A.: Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. In: Proc of the 16th Annual Network and Distributed System Security Symposium (NDSS). (2009)
24. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 226–239
25. Amigoni, F., Gatti, N.: A formal framework for connective stability of highly decentralized cooperative negotiations. Autonomous Agents and Multi-Agent Systems **15**(3) (2007) 253–279
26. Robert Hansen (RSnake): XSS (Cross Site Scripting) Cheat Sheet. `http://ha.ckers.org/xss.html` (June 2009)
27. Robert Hansen (RSnake): SQL Injection cheat sheet. `http://ha.ckers.org/sqlinjection/` (June 2009)
28. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys (CSUR) **41**(3) (2009) 15
29. Cretu-Ciocarlie, G.F., Stavrou, A., Locasto, M.E., Stolfo, S.J.: Adaptive anomaly detection via self-calibration and dynamic updating. In: RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, Berlin, Heidelberg, Springer-Verlag (2009) 41–60
30. Maggi, F., Robertson, W., Kruegel, C., Vigna, G.: Protecting a moving target: Addressing web application concept drift. In: RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, Berlin, Heidelberg, Springer-Verlag (2009) 21–40
31. Rehák, M., Staab, E., Fusenig, V., Pěchouček, M., Grill, M., Stiborek, J., Bartoš, K., Engel, T.: Runtime monitoring and dynamic reconfiguration for intrusion detection systems. In: RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, Berlin, Heidelberg, Springer-Verlag (2009) 61–80