Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 7 (2007) 547–553
Applied Soft Computing
www.elsevier.com/locate/asoc

Vulnerability analysis of immunity-based intrusion detection systems using genetic and evolutionary hackers

Gerry Dozier [a,*], Douglas Brown [b], Haiyu Hou [a], John Hurley [c]

[a] Department of Computer Science & Software Engineering, Auburn University, AL 36849-5347, United States
[b] Department of Computer Science, Clark-Atlanta University, Atlanta, GA 30314, United States
[c] Distributed Systems Integration, The Boeing Company, Seattle, WA 98124, United States

Received 7 March 2005; accepted 12 May 2006
Available online 5 October 2006

Abstract

Artificial immune systems (AISs) are biologically inspired problem solvers that have been used successfully as intrusion detection systems (IDSs). In this paper, we compare a genetic hacker with 12 evolutionary hackers based on particle swarm optimization (PSO) that have been effectively used as vulnerability analyzers (red teams) for AIS-based IDSs. Our results show that the PSO-based red teams that use Clerc's constriction coefficient outperform those that do not. Our results also show that the three types of red teams (genetic, basic PSO, and PSO with the constriction coefficient) have distinct search behaviors that are complimentary.

Keywords: Artificial immune systems; Intrusion detection systems; GENERTIA

1. Introduction

Intrusion detection [14–17,24,25,27,28] can be viewed as the problem of classifying network traffic as normal (self) or abnormal (non-self). Researchers in this area have developed a variety of intrusion detection systems (IDSs) based on: statistical methods [24,25], neural networks [3], decision trees [2], and artificial immune systems (AISs) [1,9,10,6–8,14–16,20–22,29,32]. One of the primary objectives of machine learning is to develop a hypothesis that has low error and generalizes well to unseen instances [26]. There are two types of error associated with the hypotheses developed by any learning algorithm [25,26]: false positives, known as type I errors, and false negatives, referred to as type II errors. In the context of network security, type II errors represent 'holes' [15] in an IDS.

Since type II errors do exist in IDSs, a dilemma associated with IDS design, development, and deployment is, ''Does one try to identify and/or patch holes in advance?'' Or ''Does one allow the *hackers* to identify the holes and only then try to patch them?'' In this paper, we demonstrate how GENERTIA red teams (GRTs), in the form of genetic and particle swarm search [5,23], can be used to discover holes in IDSs. This information can then be used by the designers of IDSs to develop patches or it can be used by an IDS to 'heal' itself. This research is motivated by the fact that cyber-terrorists are now turning towards automated agent-based warfare [17,25,30]. The GRT can be seen as a 'white-hat' hacker agent.

The GRTs presented in this paper are part of a larger system named GENERTIA [6–8,20,22] which contains two subsystems based on evolutionary computation [5]: a GENERTIA blue team (GBT) and a GRT. The objective of the GBT is to design AIS-based IDSs that have high attack detection rates, low error rates, and use a minimal number of detectors. The objective of the GRT is to analyze IDSs and provide feedback to the GBT. Fig. 1 shows the architecture of GENERTIA for a host-based IDS. The GBT is used to design an IDS based on input from the network manager. After a preliminary host-based IDS has been designed, the GRT performs a strength and vulnerability analysis of the IDS, based on the input specifications of the network manager. The GRT analysis results in information concerning the relative strength of detectors (labeled as RSD in Fig. 1) comprising the IDS as well as a list of vulnerabilities (holes in the IDS). This information is then given to the GBT to be used to re-design the IDS.

* Corresponding author. Tel.: +1 334 844 6327; fax: +1 334 844 6329.
E-mail addresses: doziegv@auburn.edu (G. Dozier), douglasbrown1982@yahoo.com (D. Brown), houhaiy@auburn.edu (H. Hou), john.s.hurley@boeing.com (J. Hurley).

1568-4946/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2006.05.001

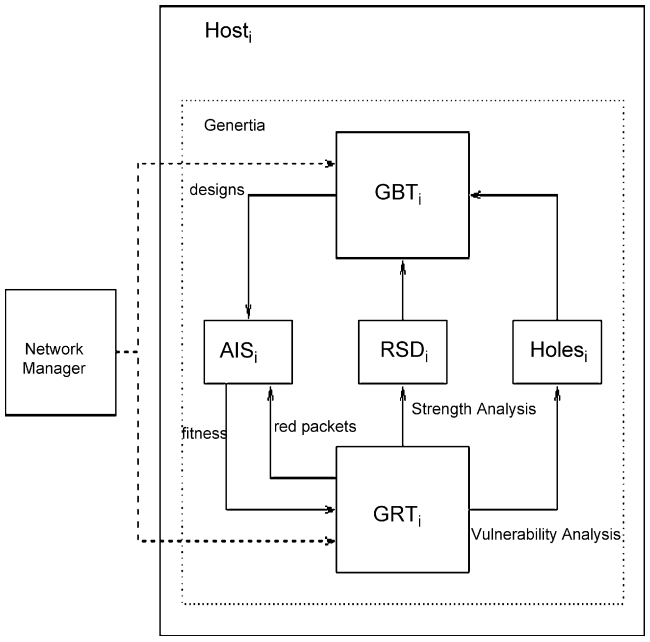

Fig. 1. Architecture of GENERTIA for a host-based IDS.

The GENERTIA concept represents a novel approach to IDS design, re-design, and maintenance. Using GENERTIA, IDSs can be analyzed for vulnerabilities before they are deployed. Genertia-based IDSs also have the ability to 'heal' themselves when vulnerabilities are discovered.

2. AIS-based intrusion detection systems

A number of researchers [1,9,15,16] have developed artificial immune systems (AISs) for networks in an effort to protect them from malicious attacks. A typical AIS-based IDS attempts to classify network traffic as either self or non-self by allowing each host to maintain and evolve a population of detectors. The evolutionary process of these detectors is as follows.

Initially, for each host, a randomly generated set of immature detectors is created. These detectors are exposed to normal network traffic (self) for a user-specified amount of time (where time is measured in packets), t_{immature}. If an immature detector matches a self-packet then the detector dies and is removed.[1] This process of removing immature detectors that match self-packets is referred to as negative selection [14,15].

All immature detectors that survive the process of negative selection become mature detectors. Thus, mature detectors will typically match non-self-packets. Mature detectors are also given a user-specified amount of time, t_{mature}, to match m_{mature} non-self-packets. This time represents the learning phase of a detector [14,15]. Mature detectors that fail to match m_{mature} non-self-packets during their learning phase die and are removed from the detector population. If a mature detector matches m_{mature} non-self-packets during its learning phase, a primary response (alarm) is invoked.

Once a mature detector sounds an alarm, it awaits a response from the network administrator to verify that the detector has identified a valid attack on the system. This response is referred to as co-stimulation [14,15]. If co-stimulation does not occur within a prescribed amount of time the mature detector will die and be removed from the detector population. Those detectors that receive co-stimulation are promoted to being memory detectors and are assigned a longer lifetime of t_{memory}. Memory detectors invoke stronger (secondary) responses when they match at least m_{memory} non-self-packets (where usually $m_{\text{mature}} > m_{\text{memory}}$).

2.1. Advantages offered by AIS-based IDSs

There are a number of advantages to using AIS-based intrusion detection. One advantage is that they provide a form of passively proactive protection via negative selection. This enables an AIS-based IDS to detect novel attacks. Another advantage is that AISs are systems that are capable of adapting to dynamically changing environments. As the characteristics of self-traffic change over time, a properly tuned AIS will be able to effectively adapt to the dynamically changing definition of self. Finally, the detectors evolved by AIS-based IDSs can easily be converted into Snort or tcpdump filters. This is especially true when constraint-based detectors are used [21,22,32]. These constraint-based detectors are easy to understand and can provide network administrators with important forensic information when new attacks are detected.

2.2. A disadvantage of using AIS-based IDSs

A disadvantage with the use of AIS-based IDSs[2] is that, at present, there is no way to know exactly what types of attacks will pass through undetected. However, a number of techniques have been developed to reduce the size and number of potential holes [10,15] but none of these can be used to alert the designer or users as to the types of attacks that will go undetected by the AIS.

In [15], Hofmeyr and Forrest use a randomly generated permutation mask for each detector in an effort to reduce the number of holes in their AIS. According to the authors, holes in their AIS result from the symmetry and fixed specificity of their r-contiguous bits matching rule. The permutation mask effectively reorders the presentation of the input pattern to each of the detectors. Therefore, a detector matching r-contiguous bits of the mutated pattern typically will not be r-contiguous bits within the pre-mutated pattern.

In [10], Dasgupta and Gonzalez propose a real-valued representation to characterize the self-space and generate detectors to cover the complementary self-space. In their

[1] Any time a detector is removed from the detector population it is automatically replaced with a randomly generated immature detector. This keeps the size of the detector population constant.

[2] Actually, this is the case with all IDSs that operate by building a model of self/non-self.

approach, the pattern space is normalized into an n-dimensional space $[0, 1]^n$. Self-space is defined as hyperspheres, with each defined by a center and a radius; while detectors are represented as hypercubes, defined by lower and upper values on each dimension. A genetic algorithm is then used to design good detectors that have large coverage and small overlap with self-space. With different values of radii, multi-level detectors representing different deviation levels are obtained to model the non-crisp boundary between self and non-self. In the next section, we demonstrate how quickly a GRT can discover holes in an AIS-based IDS.

3. The GENERTIA AIS

As stated earlier, the purpose of the GBT is to develop an AIS in the form of a set of detectors to catch new, previously unseen attacks. The purpose of the GRT is to discover holes in the AIS. As shown in Fig. 1, the AIS communicates with the GRT by receiving 'red' packets in the form of attacks from the GRT and returning the percentage of the detector set that failed to detect the 'red' packet, referred to as the fitness of the 'red' packet. The preliminary results presented in this paper are based on a single host-based IDS. However, these results can be generalized to a network where each host has an instance of GENERTIA running on it.

3.1. Representation of packets

For our AIS, packets are represented as triples (which we will refer to as data triples) of the form (ip_address, port, src), where ip_address represents the IP address of the remote host, port represents the port number of the receiving host, and src is assigned to 0 if the packet is incoming or 1 if the packet is outgoing.

3.2. The representation and behavior of detectors

The AIS maintains a population of constraint-based detectors of the form: ($lb_0, \ldots, ub_0, lb_1, \ldots, ub_1, lb_2, \ldots, ub_2, lb_3, \ldots, ub_3, lb_{port}, \ldots, ub_{port}$, src),[3] where the first four intervals represent a set of IP addresses, the fifth interval represents the lower and upper bounds on the port, and where src is 0 to denote that the detector should be used on incoming traffic or 1 to denote that the detector should be used on outgoing traffic.

An any r intervals matching rule [21,22,32] is used to determine a match between a data triple and a detector. That is, if any r numbers representing a data triple fall within the corresponding r intervals of a detector then that detector is said to match the data triple. For the experiments presented in this paper, $r = 3$.

[3] Notice that detectors based on the above representation can be viewed as constraints. Thus, a detector population can be viewed as a population of constraints where self corresponds to a set of all solutions (data triples) that satisfies the constraints and where non-self corresponds to the set of all solutions that violate at least one constraint. Therefore the intrusion detection problem can be viewed as a distributed constraint satisfaction problem [31].

If an immature detector matches a self-packet then it is first relaxed by splitting it into two parts based on where the self-packet intersects an interval and randomly removing either the lower or the upper part. This relaxation method is based on the split-detector method [32].

In our experiments, if an immature detector fails to match a self-packet after being exposed to $t_{\text{immature}} = 200$ self-packets then it becomes a mature detector and is given a life time, t_{mature}, that insures its survival for the remainder of an experiment. If a mature detector matches a self-packet it is relaxed using the split detector method. For our experiments, the values for m_{mature} and m_{memory} were set to 1. Mature and memory detectors were allowed to survive throughout the duration of an experimental run.

4. Genetic and swarm-based red teams

The GRTs compared in this paper come in the form of a steady-state GA and 12 variants of particle swarm optimization [23]. Each of the 13 GRTs evolved a population of 300 data triples. The fitness of a GRT data triple was simply the percentage of detectors that failed to detect it. For each cycle of the GA-based GRT, the worst fit data triple was replaced by an offspring if the offspring had a better fitness. An offspring was created by selecting two parents from the GRT population using binary tournament selection [13] and mating the parents using the BLX-0.5 crossover operator [11]. By using a steady-state GRT, the 300 best 'red' data triples will always remain in the population.

The swarm-based GRTs evolved a swarm of 300 particles where each particle contained: (a) a p-vector that recorded the best 'red' data triple that it has ever encountered, (b) a p-fitness value which represents the fitness of the p-vector, i.e. the percentage of the detectors of an AIS that failed to detect it, (c) an x-vector that recorded the current data triple that particle was visiting, (d) an x-fitness that recorded the fitness of the x-vector, and (e) a v-vector (velocity vector) which when added to the x-vector results in a new candidate 'red' data triple. Six of the swarm-based GRTs were instances of the canonical PSO [4] where new candidate solutions were created as follows:

$$v_{id} = v_{id} + \eta_c\varphi_c(p_{id} - x_{id}) + \eta_s\varphi_s(p_{gd} - x_{id});$$

$$x_{id} = x_{id} + v_{id}$$

where v_{id} represents the dth component of the ith particle's velocity vector, η_c and η_s represent the learning rates for the cognition and social components, φ_c and φ_s represent random numbers within $[0, \ldots, 1]$ for the cognition and social components, and g represents the index of the particle with the best p-fitness in the neighborhood of particle i. The other six swarm-based GRTs were instances of the canonical PSO using the constriction coefficient (CC) [23] where new candidate solutions were created as follows:

$$v_{id} = k(v_{id} + \eta_c\varphi_c(p_{id} - x_{id}) + \eta_s\varphi_s(p_{gd} - x_{id}));$$

$$x_{id} = x_{id} + v_{id}$$

Table 1
The distinctions of the swarm-based GRTs in terms of neighborhood, PT, and RB

Algorithm	CC	Neighborhood	PT	RB
SW0	No	Local	No	No
SW0+	No	Local	Yes	No
SW1	No	Global	No	No
SW2	No	Global	No	Yes
SW3	No	Global	Yes	No
SW4	No	Global	Yes	Yes
ccSW0	Yes	Local	No	No
ccSW0+	Yes	Local	Yes	No
ccSW1	Yes	Global	No	No
ccSW2	Yes	Global	No	Yes
ccSW3	Yes	Global	Yes	No
ccSW4	Yes	Global	Yes	Yes

where $k = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}$ represents the CC and $\varphi = \varphi_c + \varphi_s$, $\varphi > 4$.

Based on the suggestions of [4] the 12 swarm-based GRTs use asynchronous updating. Thus, each time a particle is updated the newly form x-vector is submitted to the AIS-based IDS where its x-fitness is assigned a value. The values of η_c and η_s were set to 2.3 and 1.8 according to [4].

The distinctions of the 12 swarms (denoted SW0, SW0+, SW1, SW2, SW3, SW4, ccSW0, ccSW0+, ccSW1, ccSW2, ccSW3, ccSW4) are based on: whether the CC is used, neighborhood (local, global), whether particles terminate their search when they discover a vulnerability, which we referred to as particle termination (PT), and whether the best particle, g, used for updating a particular particle is randomly selected or is the best particle with the lowest index (this only applies to those swarms that use a global neighborhood). This distinction is denoted, RB, for random best. The particles are arranged in a ring topology with a local neighborhood for a particle consisting of the particles adjacent to it. Table 1 shows the distinctions in terms of neighborhood, PT, and RB.

5. Experiment

5.1. Experiment set-up

Our training and test sets were obtained from the 1998 MIT Lincoln Lab data. The Lincoln Lab data represents 35 days of simulated network traffic for a class B network. To obtain a host-based set, we extracted packets involving host 172.16.112.50 only. From this data, we converted each packet into data triples and filtered out all duplicates and data triples involving port 80. The ports were mapped into 70 distinct ports according to [14]. We then extracted the normal traffic to form the training set. Our final training set consisted of 112 self-data triples. Our AISs were trained on approximately 80% of the training set, 89 self-data triples. The other 23 self-data triples were used to test for false positives (type I errors). Our test set consisted of all attacks launched during the 35-day period. This test set consisted of a total of 1604 data triples.

Using the above training and test sets we conducted a comparison of the 13 GRTs. Initially, an AIS was developed using a detector population size of 400. The training of the AIS consisted of selecting a data triple from the training set (self-set) and exposing the detector population to it. This process was repeated 400 times. After an AIS was trained, it was exposed to the test set. After the AIS was developed, each GRT, evolving a population of 300 malicious ('red') data triples, was allowed to interact with the AIS in order to discover holes in the system. A total of 5000 data triples were evaluated. This process was repeated 10 times.

5.2. Results

Table 2 shows the average performance of the GRTs described above in terms of the number of total number of holes discovered, the number of duplicates, and the number of distinct holes. The 10 AIS-based IDSs had an average detection rate of 0.747.

In Table 2, one can see that the GA outperforms all of the swarms with respect to distinct number of holes discovered. Of the 12 swarms, ccSW4 has the next overall best performance. In terms of the CC, the swarms that used the constriction coefficient along with particle termination (ccSW3 and ccSW4) had better performance than those that did not (SW3 and SW4).

In terms of neighborhood, when the CC is not used, the swarms with local neighborhoods (SW0 and SW0+) outperform the swarms that use a global neighborhood. However, when the constriction coefficient is used the swarms that use a global neighborhood outperform the swarms that use a local neighborhood.

In terms of PT, those swarms that used PT along with CC performed better in terms of the average number of distinct holes. This shows that terminating the motion of a particle once it has discovered a hole improves the performance of swarm search. When a particle is terminated it allows other particles in the swarm to have an increased number of trials. One can see in Table 2 that the swarms that used PT found a greater number of holes than those that did not; however, these swarms also had the greatest number of duplicates as well. When the CC was not used, the swarms that used PT had better performance in terms of the total number of holes discovered but not in terms of the

Table 2
Comparison of the seven GRTs on the 10 AIS-based IDSs with an average detection rate of 0.747 and an average false positive rate of 0.4

Algorithm	Holes	Duplicates	Distinct
GA	300.0	4.9	295.1
SW0	271.7	8.4	263.3
SW0+	298.4	21.0	277.4
SW1	297.8	51.7	246.1
SW2	292.5	50.2	242.3
SW3	299.1	62.7	236.4
SW4	300.0	62.4	237.6
ccSW0	129.7	1.0	128.7
ccSW0+	275.0	3.0	273.0
ccSW1	272.8	27.8	244.1
ccSW2	284.6	50.5	234.1
ccSW3	296.1	25.4	270.7
ccSW4	297.1	16.7	280.4

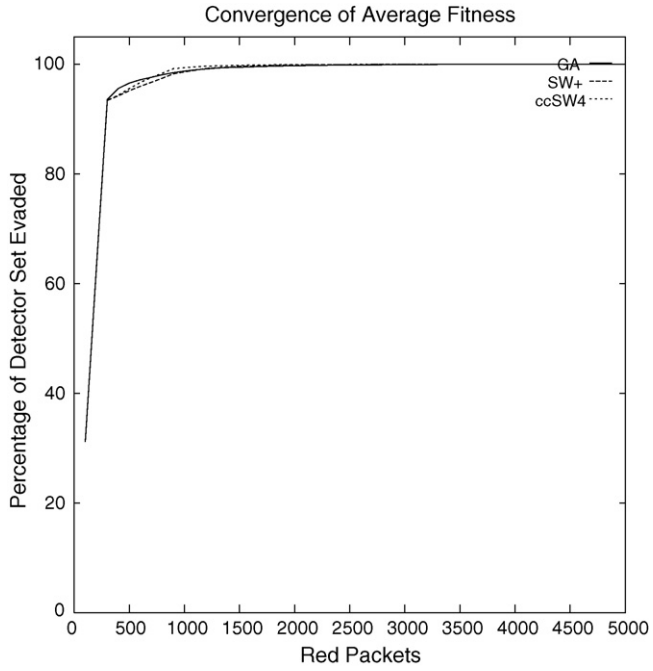

Fig. 2. Visualization of the convergence behavior in terms of average fitness of the GA, SW0+, and SW4 GRTs evolving a population size of 300 data triples.

average number of distinct holes. In this study, the use of RB did not seem to provide a performance improvement.

Figs. 2 and 3 show the convergence rates in terms of the average fitness of the population and the best fitness within the population of the GA, SW0+, and ccSW4 GRTs, the best GRTs. In Fig. 2, the average fitness of the populations increases rapidly from 30% to 95% within 500 evaluations (of red packets). After this point, the algorithms slowly converge on an average fitness that is close to 100%.

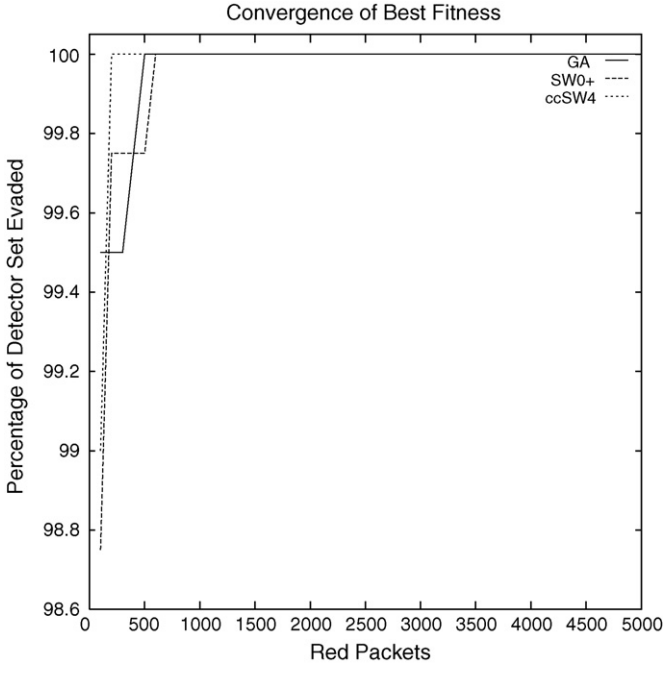

Fig. 3. Visualization of the convergence behavior in terms of best fitness of the GA, SW0+, and SW4 GRTs evolving a population size of 300 data triples.

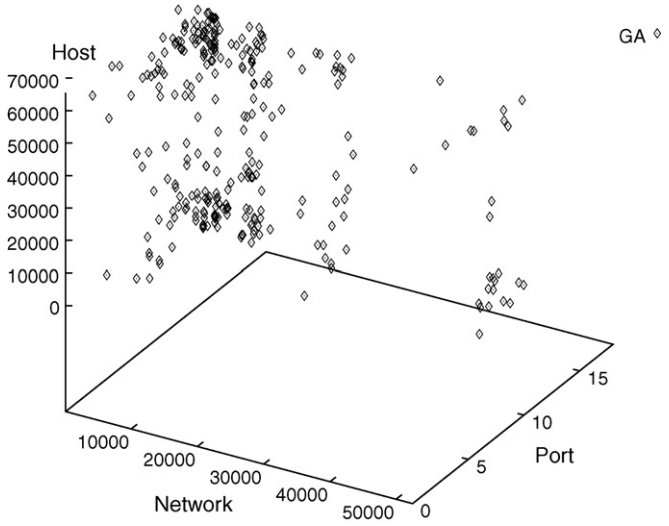

Fig. 4. Visualization of the holes evolved by the GA-based GRT.

Fig. 3 shows the convergence rates of the GA, SW0+, and SW4 GRTs with respect to the best fitness within a population. One can see that all of the algorithms start with an individual in the population that is capable of evading at least 98% of the 400 detectors. Each of the three algorithms finds their first hole in less than 1000 evaluations. Notice that ccSW4 find the first vulnerability within 250 iterations.

5.3. Visualization of vulnerabilities

Figs. 4–6 show a 3D visualization of the holes where the *x*-axis represents the network, the *y*-axis represents the host and the *z*-axis represents the port. In this visualization we assume

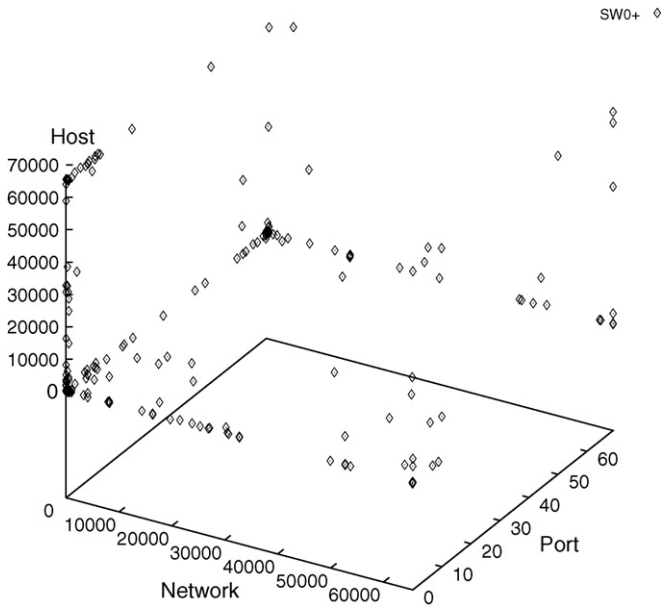

Fig. 5. Visualization of the holes evolved by the SW0+-based GRT.

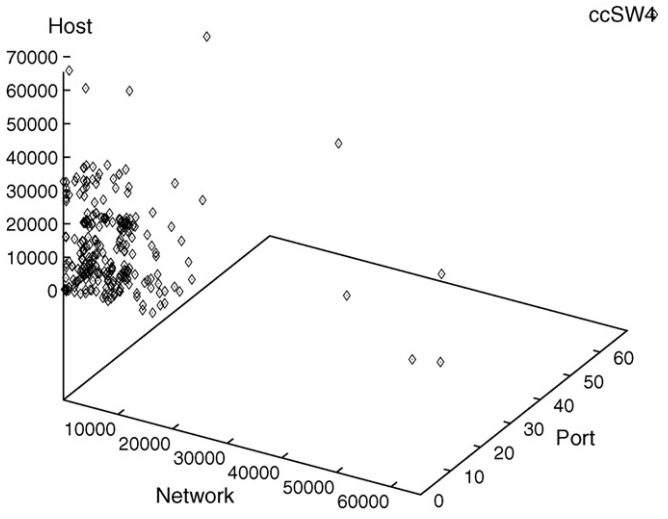

Fig. 6. Visualization of the holes evolved by the SW4-based GRT.

that the red packets are coming from a class B network. In Figs. 4–6, one can see that the GA and the swarms have a completely different search behavior. The GA tends to find solutions in clusters while SW0+ seems to discover holes at the boundaries of the search space. This is interesting because based on our representation of a detector (constraint based intervals) the extreme values of source and host IP addresses and extreme values of port numbers are the hardest to cover. Based on Fig. 5 one can conclude that a better form of detector would one that uses wrap around intervals. This discovery has lead to the development of constraint-based detectors with wrap around intervals [20].

In Fig. 6, ccSW4 concentrates in one area where is was able to discover a large cluster of vulnerabilities. The results seen in Figs. 4–6 suggest the possibility that a hybrid GA/PSO algorithm, one which we refer to as a genetic swarm, may be more effective than either GA or PSO alone. Upon further reflection on Figs. 5 and 6, it seems as though one could develop larger clusters by adapting the neighborhood size during search.

6. Conclusions and future work

In this paper, we compare a number of evolutionary red teams for discovering vulnerabilities of immunity-based intrusion detection systems. The overall best performer was the GA followed by PSOs: ccSW4 and SW0+. Those PSOs using particle termination along with the constriction coefficient performed better than those PSOs that did not use both. When visualizing the vulnerabilities discovered by the three best red teams (GA, SW0+, and ccSW4), one can see that they find different types of vulnerabilities in entirely different areas of the IP address space. This variation in search behavior can be considered as an asset points in the direction of the development of hybrid red teams composed of a number of different evolutionary computations. This will be a direction of future research.

References

[1] J. Balthrop, S. Forrest, M. Glickman, Revisiting LISYS: parameters and normal behavior, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), IEEE Press, 2002.
[2] E. Bloedorn, A.D. Christiansen, W. Hill, C. Skorupka, L.M. Talbot, J. Tivel, Data Mining for Network Intrusion Detection: How to Get Started, The MITRE Corporation, 2001 available at: http://www.mitre.org/support/papers/archive01.shtml (August 2001).
[3] J. Cannady, Artificial neural networks for misuse detection, in: Proceedings of the 1998 National Information Systems Security Conference (NISSC'98), October 5–8, Arlington, VA, (1998), pp. 443–456.
[4] A. Carlisle, G. Dozier, An off-the-shelf PSO, in: Proceedings of the 2001 Workshop on Particle Swarm Optimization, Indianapolis, IN, 2001 , pp. 1–6.
[5] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991 .
[6] G. Dozier, D. Brown, J. Hurley, K. Cain, Vulnerability analysis of immunity-based intrusion detection systems using evolutionary hackers, in: The Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO-2004), LNCS 3102, June, Seattle, WA, 2004, pp. 263–274 (Springer; best paper award).
[7] G. Dozier, D. Brown, J. Hurley, K. Cain, Vulnerability analysis of AIS-based intrusion detection systems via genetic and particle swarm red teams, in: The Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004), June 19–23, Portland, OR, (2004), pp. 111–116.
[8] G. Dozier, IDS vulnerability analysis using GENERTIA red teams, in: The 2003 International Conference on Security and Management (SAM'03), June 23–26, Las Vegas, Nevada, USA, (2003), pp. 171–176.
[9] D. Dasgupta, An overview of artificial immune systems and their applications, in: D. Dasgupta (Ed.), Artificial Immune Systems and Their Applications, Springer, 1999, pp. 3–21.
[10] D. Dasgupta, F. Gonzalez, An immunity-based technique to characterize intrusions in computer networks, IEEE Trans. Evol. Comput. 6 (3) (2002) 281–291 (IEEE Press).
[11] L. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval-Schemata, in: L. Darrell Whitley (Ed.), Proceedings of the 1992 Foundations of Genetic Algorithms (FOGA-2), Morgan Kaufmann, 1992, pp. 187–202.
[13] D.E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in: Gregory J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann Publishers, 1989 , pp. 69–93.
[14] S. Hofmeyr, An Immunological Model of Distributed Detection and Its Application to Computer Security, PhD Dissertation, Department of Computer Science, The University of New Mexico, 1999.
[15] S. Hofmeyr, S. Forrest, Immunity by design: an artificial immune system, in: The Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO-1999), Morgan Kaufmann, San Francisco, 1999, pp. 1289–1296.
[16] S. Hofmeyr, S. Forrest, Architecture for an artificial immune system, Evol. Comput. 7 (1) (1999) 45–68.
[17] Honeynet Project, Know Your Enemy: Revealing the Security Tools, Tactics and Motives of the Blackhat Community, Addison-Wesley, 2002.
[20] H. Hou, G. Dozier, Comparing the performance of binary-coded and constraint-based detectors, in: Proceedings of the 2004 Congress on Evolutionary Computation, June 19–23, Portland, OR, (2004), pp. 772–777.
[21] H. Hou, Artificial Immunity for Computer Network with Constraint-Based Detector, Masters Thesis, Department of Computer Science and Software Engineering, Auburn University, 2002.
[22] H. Hou, J. Zhu, G. Dozier, Artificial immunity using constraint-based detectors, in: Proceedings of the 2002 World Automation Congress (WAC'02), vol. 13, TSI Press, 2002, pp. 239–244.
[23] J. Kennedy, R. Eberhart, Swarm Intelligence, Morgan Kaufmann, 2001.

[24] D.J. Marchette, A statistical method for profiling network traffic, in: Proceedings of the USENIX Workshop on Intrusion Detection and Network, 1999, pp. 119–128.
[25] D.J. Marchette, Computer Intrusion Detection and Network Monitoring: A Statistical View-Point, Springer, 2001.
[26] T.M. Mitchell, Machine Learning, McGraw-Hill, 1997.
[27] S. Northcutt, J. Novak, Network Intrusion Detection: An Analyst's Handbook, 2nd ed., New Riders, 2001.
[28] S. Northcutt, M. Cooper, M. Fearnow, K. Frederick, Intrusion Signatures and Analysis, New Riders, 2001.
[29] A. Somayaji, S. Hofmeyr, S. Forrest, Principles of a Computer Immune System, The New Security Paradigm Workshop, 1997, pp. 75–82.
[30] A.J. Stewart, Distributed Metastasis: A Computer Network Penetration Methodology, 1999, available at: http://www.citeseer.nj.nec.com/387640.html.
[31] M. Yokoo, Distributed Constraint Satisfaction, Springer-Verlag, 2001.
[32] J. Zhu, Use of an Immune Model to Improve Intrusion Detection on Dynamic Broadcast Local Area Networks, Masters Thesis, Department of Computer Science & Software Engineering, Auburn University, 2002.