ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2008, Vol. 37, No. 2

GENETIC ALGORITHM BASED INTERNET WORM PROPAGATION STRATEGY MODELING

Nikolaj Goranin, Antanas Čenys

Information Security Laboratory, Faculty of Fundamental Sciences, Vilnius Gediminas Technical University
Saulėtekio al. 11, SRL-I-415, LT-10223, Vilnius, Lithuania

Abstract. Existing malware propagation models mainly concentrate on malware epidemic consequences modeling, i.e. forecasting the number of infected computers, and are based only on current malware propagation strategies. In this article we propose a genetic algorithm based model, which aims at evaluating existing as well as modeling other potentially dangerous Internet worms' propagation strategies. The efficiency of strategies is evaluated by applying the proposed fitness function. Genetic algorithm is selected as a modeling tool taking into consideration the efficiency of this method while solving optimization and modeling problems with large solution space. The main application of the proposed model is a countermeasures planning in advance and computer network design optimization.

1. Introduction

According to [17], almost 3/5 of all companies in the United Kingdom have faced different kinds of information security breaches in 2006, almost 50% of which were caused by malware, i.e. viruses, worms, Trojans, etc, i.e. software that was created with the aim to harm computer software or to infect it without the permission and knowledge of a legal user [16]. In Lithuania according to The Communications Regulatory Authority of the Republic of Lithuania [14], approximately 70% of information security breaches that affected companies and home users were caused by malware. In 2006 the total number of new malicious programs was up 41% from 2005 [2] and even up to 172% according to Corrons research [13]. One more 2006 year trend marked by Corrons [13] is the use of worms as a means for propagation of other malware. Data for year 2007 are incomplete but, according to [7], prediction at a 60% percent increase from 2006 in unique malware is expected. There have not been any new large-scale worms targeting Windows services since 2005. On the other hand, vulnerabilities found in anti-virus, backup or other application software, can result in worms later. Most notable in 2007 was the worm exploiting the Symantec antivirus buffer overflow flaw [20].

Despite the different percent presented by different antivirus companies [2, 13] and incomplete data for 2007 it is obvious that the rate of malware usage by e-criminals has the tendency to increase and protection against it is a crucial task. Worms remain a significant part of all malware and may be defined as one of the most potential threats in 2008. Worms are network viruses, primarily replicating on networks. Usually a worm will execute itself automatically on a remote machine without any extra help from a user. However, there are worms, such as mailer or mass-mailer worms, that will not always automatically execute themselves without the help of a user [16]. In this article we analyze and model Internet worm propagation strategies, since their replication mechanisms differ significantly from mailer and mass-mailer worms. Propagation of most worms is rapid (compared with classical computer viruses) and aggressive. Worms such as Code Red and Nimda have been persistent for longer than 8 months since their introducetion date. As worms spread through nearly all networks, they find nearly all of the weakest hosts accessible and begin their lifecycle anew on these systems. This then gives worms a broad base of installation from which to act [10].

Prior information security analysis techniques are not effective in evaluating worms. The main issues faced in worm evaluation include the scale and propagation of the infections [10]. Modeling allows Internet worm researchers to predict damage for a new worm threat [4], understand the behavior of malware, including spreading characteristics [15], understand the factors affecting the malware spread, determine the required effectiveness of countermeasures in order to control the spread and facilitate network designs that are resilient to malware attacks [12], predict the failures of the global network infrastructure [8]. Our proposed model allows evaluating existing and modeling other potentially dangerous Internet worms' propagation strategies. The main application of a proposed model is a countermeasures planning in advance and computer network design optimization. Genetic algorithm [9] was selected as a modeling tool since it simulates natural selection by means of repeatedly evolving population of solutions (malware propagation strategies in our case) and therefore may be used for predicting and modeling possible future propagation strategies. Genetic algorithm modeling has been proved to be effective in many areas such as business decision making, bioinformatics and other [1, 3, 11, 18].

2. Current worm propagation strategies

We define the Internet worms propagation strategy as a combination of methods and techniques, used by the worm to achieve tasks assigned to it by the worm creator. So the strategy suitable to achieve one specific task (e.g., creating the botnet [22]) may be not useful for another (e.g., disrupting Internet functioning). Modern worms are usually created on a modular basis and may contain all or some of the following parts [10]: a reconnaissance module, that scans the Internet for vulnerable hosts; an attack module, that may exploit from one to many known vulnerabilities at potentially vulnerable host; a communication module that allows worms to communicate between themselves or to transfer information to the worm management center; a command module, that allows to accept commands; and an intelligence module, that insures functioning of the communication module, since it contains information how to find a neighbor worm for communication. Specific methods used in each of the modules are called patterns and a strategy can be also defined as a combination of patterns. A strategy is also dependent on worm introduction techniques, i.e. method used to release worm to the wild, connection protocol used (e.g. TCP or UDP), etc. Since the number of existing and historic worms is high, we will describe only two propagation strategies used by CodeRed and Ramen, since they represent two different attitudes in complexity, vulnerable platform and functionality and can provide an understanding of strategies used in the wild.

On June 18th 2001 a serious Windows IIS vulnerability was discovered. On July 13th 2001 Code Red worm version 1 that exploited this single vulnerability was released. Due to a code error in its random number generator, it did not propagate well. 10:00 UTC of July 19th Code Red version 2 was released with the corrected random generator. It generated 100 threads. Each of the first 99 threads randomly chose one IP address and tried to set up connection on port 80 with the target machine (if the system was an English Windows 2000 system, the 100th worm thread would deface the infected system's web site, otherwise the thread was used to infect other systems, too) [4]. The worm was programmed to scan hosts in /8 with a 50% probability, /16 – with 37.5% probability and with 12.5% probability it would scan a totally random network [10]. Subnetworks 127.0.0.0/8, loopback, 224.0.0.0/8, multicast were excluded [8]. If the connection was successful, the worm would send a copy of itself to the victim web server to compromise it and continue to find another web server. If the victim was not a web server or the connection could not be setup, the worm thread would randomly generate another IP address to probe. The timeout of the Code Red connection request was programmed to be 21 seconds. Netcraft web server survey showed that there were about 6 million Windows IIS web servers at the end of June 2001 [4]. More than 350.000 of them were infected in several hours [19].

The Ramen worm appeared in January 2001. Ramen attacked RedHat Linux 6.0, 6.1, 6.2, and 7.0 installations, taking advantage of the default installation and three known vulnerabilities: FTPd string format exploits against wu-ftpd 2.6.0, RPC.statd Linux unformatted strings exploits, and LPR string format attacks. This vulnerable software could be installed on any Linux system, meaning the Ramen worm can affect other Linux systems, as well. The worm acted in the following way: defaced any Web sites it found; disabled anonymous FTP access to the system; disabled and removed the vulnerable rpc.statd and lpd daemons, and ensured the worm would be unable to attack the host again; installed a Web server on TCP port 27374, used to pass the worm payload to the child infections; removed any host access restrictions and ensured that the worm software would start at boot time; notified the owner (worm creator) of two e-mail accounts of the presence of the worm infection. Worm then began scanning for new victim hosts by generating random class B (/16) address blocks (scans were restricted from 128/8 to 224/8, the most heavily used section of the Internet). Web server acted as a small command interface with a very limited set of possible actions. The mailboxes served as the intelligence database, containing information about the nodes on the network. This allowed the owners of the database to be able to contact infected systems and operate them as needed [10].

3. Prior and related work

The Random Constant Spread (RCS) model [19] was developed by Staniford et al. using empirical data derived from the outbreak of the *CodeRed* worm. It assumes that the worm has a good random number generator that is properly seeded. The model assumes that a machine cannot be compromised multiple times and operates several variables: K is the constant average compromise rate, which is dependant on worm processor speed, network bandwidth and location of the infected host; $a(t)$ is the proportion of vulnerable machines which have been compromised at the instant t, $Na(t)$ is the number of infected hosts, each of which scans other vulnerable machines at a rate K per unit of time. But since a portion $a(t)$ of the vulnerable machines is already infected, only $K \cdot (1-a(t))$ new infections will be generated by each infected host, per unit of time. The number n of machines that will be compromised in the interval of time dt (in which a is assumed to be constant) is thus given by:

$$n = (Na) \cdot K(1-a)dt. \tag{1}$$

N is assumed to be a large constant address space so the chance that the worm would hit the already infected host is negligible. From this hypothesis, $n=d(Na)=Nda$. It is also possible to write

$$Nda = (Na) \cdot K(1-a)dt. \tag{2}$$

From this we have

$$\frac{da}{dt} = Ka(1-a), \tag{3}$$

where

$$a = \frac{e^{K(t-T)}}{1+e^{K(t-T)}}. \tag{4}$$

So the model can predict the number of infected hosts at time t if K is known. The higher is K, the quicker the satiation phase will be achieved by worm. As [10] states, although more complicated models can be derived, most network worms will follow this trend. So in our article we will use this model to obtain a measure of the growth rate of the worm which uses a specific strategy.

Chen, Gao and Kwiat [23] propose the AAWP discrete time model, in the hope to better capture the discrete time behavior of a worm. However, according to [8], continuous model is appropriate for large scale models, and the epidemiological literature is clear in this direction. The assumptions on which the AAWP model is based are not completely correct, but it is enough to note that the benefits of using a discrete time model seem to be very limited.

On the other hand, Zanero et al in [8] propose a sophisticated compartment based model, which treats Internet as the interconnection of autonomous systems, i.e. subnetworks. Interconnections are a so-called “bottlenecks”. The model assumes that inside a single autonomous system (or inside a densely connected region of an AS) the worm propagates unhindered, following the RCS model. The authors motivate the necessity of their model via the fact that the network limited worm Slammer, which was using UDP protocol for propagation ,was following the RCS model till the “bottlenecks” were flooded by its scans.

Zou et al in [4] propose a two-factor propagation model, which is more precise in modeling the satiation phase taking into attention the human countermeasures and the decreased scan and infection rate due to the large amount of scan-traffic. The same authors have also published an article on modeling worm propagation under dynamic quarantine defense [6] and evaluated the effectiveness of several existing and perspective worm propagation strategies [5].

So as described above, all existing Internet worm propagation models concentrate on epidemic consequences modeling, i.e. forecasting the number of infected computers at some specified moment of time after the start of worm propagation, and are based on the current malware propagation strategies.

4. Propagation strategy modeling

4.1. General assumptions

Applying genetic algorithm to propagation strategy modeling and prediction seems to be a promising area, since the algorithm itself simulates the natural evolution. In the current study, we have chosen to model strategies for a theoretical Internet worm, which aims infecting the largest amount of hosts during a fixed relatively short period of time. The general algorithm is presented in Figure 1.

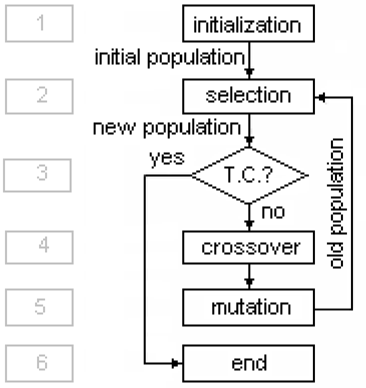

Figure 1. Algorithm flow chart

During the initialization stage (Figure 1-1) initial population of strategies is generated. Each strategy is represented as a chromosome. At selection stage (Figure 1-2) strategies are selected through a fitness-based process and in case termination condition (T.C., Figure 1-3) is not met evolutionary mechanisms are started (Figure 1-4/5). In case termination condition is reached, algorithm execution is ended (Figure 1-6).

4.2. Experiment conditions

Initial population is generated on a random basis, i.e. each individual, representing separate worm propagation strategy is combined of random genes' values. Population size N is equal to 50. Population size remains constant after each new generation. The combined termination condition was selected. The algorithm would stop producing new generations in two cases: either the number of generations have reached 100, or the fitness evaluation of the fittest individual in a population remains constant for 10 consecutive generations. The crossover point for each pair of parents is selected randomly and defines the gene, after which the crossover operation is performed. The mutation operator defines the gene of a newly generated individual that should change value from current to any other random value from the range of possible gene values. Mutation operator is activated to each newly generated individual with a 0.005 probability. Fitness proportionate selection was used. After the assignment of fitness value to an individual its selection probability is calculated according to (5), that defines the individual's probability to be selected as a parent.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \qquad (5)$$

where p_i is the probability and f_i is the fitness of individual i in the population consisting of N (50 in our case) individuals. In such a way more individuals with higher fitness value get the higher chance to leave the offspring than those with the lower one.

4.3. Strategy representation

Each strategy is represented as a chromosome, which is combined of genes, i.e. combination of techniques and methods. Genes are divided into compulsory (vitally necessary for worm to propagate, e.g. scanning gene, exploit gene or controlling non-compulsory genes) and non compulsory (i.e. giving a worm some functionality, that may or may not result in additional worm efficiency, e.g. remote administration function). Compulsory genes are active in all chromosomes. Genes that are not compulsory have an additional activation gene. Such an assumption allows modeling any combination of methods and techniques and insures the fixed length of the chromosome. Chromosome structure is described in Table 1.

Table 1. Chromosome structure

No., Gene code (A^a /NC) *Gene description*	*Range of values or sample values*	*Comments*

1 / IP_GEN (A)
Defines potential victim's IP address generation algorithm.
	…
	Random;
	Random, excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast;
	Random, excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast and LAN addresses;
	Differentiated random: /8 with *X%* probability, /16 - with *Y%* probability, *Z%* - fully random;
	Random /16 addresses in range from 128/8 to 224/8;
	Random addresses from networks reserved for home user networks (DSL, etc.);
	…
		X%, Y%, Z% - are generated randomly at initial population generation phase, in case a differentiated random algorithm was selected for generating individual representing propagation strategy.
		Since it is not possible to present all IP generation algorithms used in the experiment, only a short representative selection was presented in "Range of values" column of IP_GEN gene.
		In a chromosome representing propagation strategy only a reference number to the IP generation algorithm, stored in an external array, is provided.

2 / OS_PLATF(A) Defines the OS platform the worm can function on.	DOS, *nix (Unix, Solaris, Linux), Win 9x, Win NT (NT, 2000, XP, Vista), Apple OS.	The proposed list of values is not exhaustive. It only represents values that were used for the experiments.
3 / TRANSF (A) Defines worm's body transfer mechanism	Connectionless (also called "Fire and forget") Connection oriented	Connectionless mechanism uses UDP protocol for worm body transfer. Assumption is done, that worm can fit in one datagram. Connection oriented mechanism uses TCP protocol for worm transfer to the target host. For simplicity reasons, we make an assumption that worm can be transferred in both methods, as for example attacking DNS (53/TCP,UDP) server. Without such an assumption additional check would be necessary.
4 / EXPL_1 (A) Defines the first exploit to be included in worm's body. The first exploit is compulsory, since at least one exploit is necessary for worm's propagation.	Random exploit from the array of exploits for the selected OS platform, for example exploit based on CVE-2004-0297	Array of exploits for each platform consists of 20 exploits with "remote" exploitation feature and is based on a list of Common Vulnerabilities and Exposures (CVE) [21]. Each exploit is assigned with a random percent number (due to the lack of statistical information), which defines the part of potentially vulnerable hosts (i.e. hosts running this software type) among all hosts running this OS platform.
5, 6, 7, 8, 9, 10, 11 / EN_EXPL_N (N=2-8) (A) Exploit EXPL_N activation gene.	True/False	Enables EXPL_N gene, if gene value is True. The maximum number of exploits (including compulsory) is limited to 8 for simplicity reasons.
12, 13, 14, 15, 16, 17, 18 / EXPL_N (N=2-8) (AE) See EXPL_1 for description	See EXPL_1 for description.	See EXPL_1 for description.
19 / EN_MEM (A) MEM activation gene.	True/False	Enables MEM gene, if gene value is True.
20 / MEM (NC) Defines type of memory the worm uses.	RAM, file, DB	Memory module is used to store information about communication paths. Additional information may also be stored.

No., Gene code (A^a /NC) *Gene description*	*Range of values or sample values*	*Comments*
21 / EN_HIER (A) HIER activation gene.	True/False	Enables HIER gene, if gene value is True
22 / HIER (AE) Defines worm network hierarchy	Autonomous Centralized hierarchy; Decentralized hierarchy.	Autonomous – each infected host can act as a management source; Centralized hierarchy – there exists only one management center; Decentralized hierarchy – several management centers exist on a network.
23 / EN_COM (A) COM activation gene.	True/False	Enables COM gene, if gene value is True
24/ COM (AE) Defines worms‘ communication algorithm.	Communicate via child/parent chain; Communicate via direct connection between infected and management hosts.	In a child/parent chain each individual worm knows only the connection to its parent, i.e. the host from which it was infected and parent knows only these hosts that were infected by him. This mechanism may be used in autonomous, centralized and decentralized hierarchies. Direct communication assumes that each infected host stores information, that is used to achieve management host (in case of centralized h.), some or part of management hosts (in case of decentralized h.)
25 / EN_EXEC (A) EXEC activation gene.	True/False	Enables EXEC gene, if gene value is True
26 / EXEC (AE) Defines remote worm management features.	Standard functionality Update functionality Standard + Update functionality	Standard functionality – supports remote execution of functions, included in the worm's body or functions of a compromised host's OS; Update functionality – supports update or change of any worm module. Standard + Update – combined functionality of the two previous techniques.
27 / EN_ADD (A) ADD activation gene.	True/False	Enables ADD gene if gene value is True.
28 / ADD (AE) Defines additional worm functionality features.	Deface local host; write to MBR to remain after reboot; DDoS support; or any combination of these techniques.	Assumption is done, that additional functionality is performed prior to begging of IP generation and exploit transfer.
29 / EN_EVOL (A) EVOL activation gene.	True/False	Enables EVOL gene if gene value is True.
30 / EVOL (A) Defines worm's evolution.	… If only one exploit is used, then after 100 generations enable EXEC gene with Update functionality, enable additional exploits, update the worm's body with additional exploits. …	Evolutionary gene allows the change of worm's propagation strategy after defined number of worm generations or according to propagation conditions analysis (propagation slow-down). Since in our experiment we model worm's propagation in the initial stage, evolutionary genes do not play an important role.

a – A – always active gene, AE – active if enabled by activation gene.

As it can be seen from the table, many genes are related, for example, disabling memory gene will effect in making hierarchy and communication genes ineffective even if they were enabled. Despite this, the proposed propagation strategy representation remains universal even in case such gene combinations are created in the initial population generation phase due to the nature of genetic algorithms, since lifeless combinations will be eliminated depending on the fitness function evaluation criteria during the evolutionary process and only the most fittest will survive.

4.4. Fitness function

By definition the fitness function is a particular type of objective function that quantifies the optimality of a solution (i.e. an individual in a population) so that the particular individual may be ranked against all the other individuals. The task of our experiment is to create a worm's propagation strategy that aims infecting the largest amount of computers in a limited period of time. From [19] we can say that the propagation strategy efficiency can be evaluated by value *K,* i.e. the number of computers the first worm individual in the wild can infect in a fixed time period. That means that the higher is *K*, the higher is the fitness of a propagations strategy. Our *K* calculations by fitness function are based on combined statistical or empirical evaluation of time expenditures of strategy's functionnality and probabilistic evaluation of strategy's functionality efficiency. The fitness function we used is the following:

$$F(S) = k \cdot p_1 \cdot p_2 \cdot p_3 \sum_{i=4}^{30} p_i , \qquad (6)$$

where: S – evaluated strategy; p_1 – probability that the generated IP address exists and alive, p_2 – probability that host is running the OS platform that the worm supports, p_3 – probability that worm will be successfully transferred to the potential victim, p_i – probability that the i^{th} gene will result in an infected host, i=4..30; k – the number of cycles the worm, using the evaluated strategy, can perform in one second time interval. k is calculated according to the following expression

$$k = \frac{1}{\sum_{j=1}^{30} t_j}, \tag{7}$$

where t_j are time expenditures needed for j^{th} gene functionality, j=1..30.

So the fitness function can be read as: "Strategy S can perform k cycles per second. During each cycle the worm, using this strategy, will infect a host in case the generated IP address exists, the host is up and running the OS platform the worm supports, worm is successfully transferred to the target and any other gene results in host infection. The calculated value of the evaluated strategy is its K value." In case the i^{th} gene is not active, p and t are equal to 0. p and t for activation genes are always equal to 0. Simplification is done while evaluating exploits' probabilities: probabilities are assumed to be rather low, i.e. less than 0.05; probabilities of all exploits are independent, i.e. host can be infected only by one of the exploits. It is also obvious that genes i=19..30 do not provide additional probability to host infection (i.e. p=0) and differ only in time consumption. Inclusion of these genes in the model is done in order to provide a wide model framework, which enables evaluation of propagation strategy features, different from propagation speed, such as survivability, visibility, manageability, etc.

Since the presentation of all time expenditures and probabilistic values for all possible strategy techniques listed or referenced in Table 1 is out of topic of this article we present a sample calculation of K value for a random propagation strategy, generated during the initial population generation phase. Probabilistic and time expenditure values for the sample strategy are presented in Table 2.

S_i=(IP_GEN="Random, excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast"; OS_PLATF="Apple OS"; TRANSF="Connection oriented"; EXPL_1=" CVE-2007-3876"; EN_EXPL_2="False"; EN_EXPL_3="False"; EN_EXPL_4="False"; EN_EXPL_5="True"; EN_EXPL_6="False"; EN_EXPL_7="False"; EN_EXPL_8="False"; EXPL_2="-"; EXPL_3="-"; EXPL_4="-"; EXPL_5=" CVE-2004-0485"**; EXPL_6="-"; EXPL_7="-"; EXPL_8="-"; EN_MEM="False"; MEM="-"; EN_HIER="True"; HIER="Autonomous"; EN_COM="False"; COM="-"; EN_EXEC="True"; EXEC="Update functionality"; EN_ADD="True"; ADD="Write to MBR to remain after reboot"; EN_EVOL="False"; EVOL="-")*

* – Apple Mac OS X SMB Vulnerability

** – Apple Mac OS X URI Handler Arbitrary Code Execution Vulnerability.

Table 2. Parameter values used in fitness calculation of the sample strategy S_i

Gene Nr.	1	2	3	4	5	6	7	8	9	10
p	0.6	0.04	0.95	0.04	0	0	0	0	0	0
t	0.01	0.005	0.15	0.005	0	0	0	0	0	0
Gene Nr.	11	12	13	14	15	16	17	18	19	20
p	0	0	0	0	0.01	0	0	0	0	0
t	0	0	0	0	0.005	0	0	0	0	0
Gene Nr.	21	22	23	24	25	26	27	28	29	30
p	0	0	0	0	0	0	0	0	0	0
t	0	0.01	0	0	0	0.005	0	0.005	0	0

k=5.13; $F(S_i)$=0.0058

For comparison, the obtained K value for CodeRed v.2 worm in case it would use only one thread instead of 100 is much higher and is equal to 0.015. In case of 100 threads, K would reach 1.5, which is rather similar to the observed CodeRed v.2 K value of 1.6 [8]. Differences between calculated and observed values of K can be explained by inaccuracies in estimations of time consumption.

4.5. Experiment results, discussion and future work

While modeling the evolution of propagation strategies, that aim infecting the maximum number of hosts in a limited period of time, 10 algorithm runs were performed. The highest K value obtained was equal to 1.18 (equivalent to 118 in case of 100 threads) and corresponded to the rather simple strategy that got use of Windows platform (p=0.9), random (excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast) IP generation mechanism, connectionless UDP based transfer mechanism (p=0.8, t=0.01) and 4 exploits with rather high to high probabilities (0.05; 0.05; 0.045; 0.035). All other genes were disabled. The change in the fitness of the whole population and the fitness of the best individual in case when the highest K value was obtained, can be seen on Figure 2.

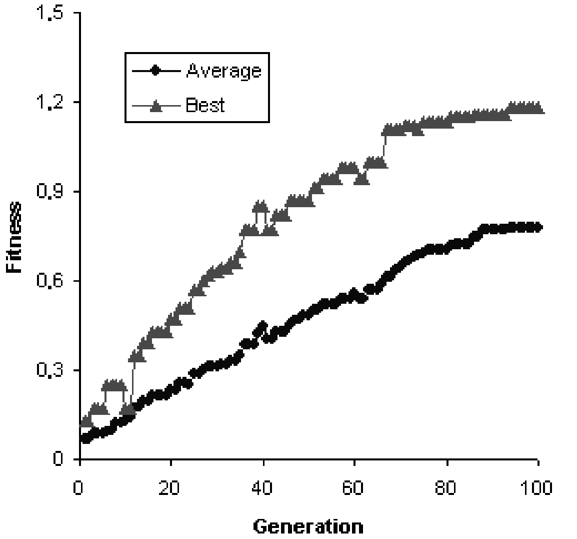

Figure 2. Change in fitness of the whole population and the best individual

Although we cannot claim that the found solution is optimal, the strategy of using simple "fire and forget" mechanism on a popular platform and a combination of several exploits with high probability seems to be very effective and challenging for countermeasure planning. Partially the proof of this concept was provided by the creators of the Slammer worm, which used only one exploit but still was much quicker than connection oriented CodeRed v.2 (Slammer doubling time was 8.5 sec. compared to CodeRed v.2 37 min. [8])

The proposed model provides a general framework for evaluating different worms' propagation strategy parameters (speed, survivability, manageability, etc.). The performed tests have shown the model's applicability. Still some improvements should be made in order to make its use more practical. In case of propagation rate at initial propagation stage evaluation, discussed in this article, conditional probabilities should be introduced for exploits in order to evaluate the cases when single host can be infected by two or more different exploits, more precise statistical evaluation of time consumption by different worm mechanisms should be made, check of genes' relations should be included into fitness function (e.g. if the specific exploit can be transferred via UDP or TCP protocol). We also plan extending the model for countermeasures planning. In that case, two co-evolving populations of propagation strategies and countermeasure application strategies will be used and the efficiency of each strategy will be evaluated against the array of opposing strategies (i.e. countermeasure application strategy's efficiency would be evaluated in measure of propagation strategy decrease rate and vice versa). But even in the current state the model can be used for predicting potential propagation strategies.

5. Conclusions

In this article the genetic algorithm based model for Internet worm's propagation strategy evolution modeling at initial propagation phase was proposed. The model consists of a propagation strategy representation structure, genetic algorithm acting under specified conditions and a fitness function, which evaluates the strategy's infection rate at the initial propagation phase, leaning on probability and time consumption estimations of strategy's used methods.

The proposed model was tested on existing worms' propagation strategies with known infection probabilities. The tests have proved the effectiveness of the model in evaluating propagation rates. The modeling of perspective propagation strategies aiming to infect the highest number of computers in a limited period of time has shown that these strategies tend to evolve to rather simple solutions, making use of a popular OS platform, quick connectionless UDP based transfer mechanisms and a combination of several (4-5) exploits with high infection probability.

The main model application area is countermeasures planning, since the model predicts the propagation strategy trends. The proposed model can be also used as a framework for evolution modeling of other parameters of propagation strategies, such as population visibility, manageability, etc., if fitness function modification is made. Several future model improvements and extensions were discussed.

References

[1] **A. Chittur.** Model Generation for an Intrusion Detection System Using Genetic Algorithms. *Internet link: http://www1.cs.columbia.edu/ids/publications/gaids-thesis01.pdf*, 2001.

[2] **A. Gostev.** Kaspersky Security Bulletin 2006: Malware Evolution. *Technical report. Kaspersky Lab*, 2007.

[3] **C. Birchenhall, N. Kastrinos, S. Metcalfe.** Genetic algorithms in evolutionary modeling. *Journal of Evolutionary Economics*, 1997, *Vol.*7, 375-393.

[4] **C.C. Zou, W. Gong, D. Towsley.** Code Red Worm Propagation Modeling and Analysis. *Proceedings of the* 9*th ACM conference on Computer and communications security*, 2002, 138-147.

[5] **C.C. Zou, W. Gong, D. Towsley.** On the performance of Internet worm scanning strategies. *Performance Evaluation*, 2005, *Vol.*63, 700-723.

[6] **C.C. Zou, W. Gong, D. Towsley.** Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense. *Proceedings of WORM'*03, 2003, 10.

[7] **C. Schmugar.** Malware estimation for 2007. *McAfee News. McAfee Avert Labs*, 2007.

[8] **G. Serazzi, S. Zanero.** Computer Virus Propagation Models. *Lecture Notes in Computer Science. SPRINGER-VERLAG*, 2004, 26-50.

[9] **J. Holland.** Adoption in natural and artificial systems. *The MIT press*, 1975, 211.

[10] **J. Nazario.** Defense and Detection Strategies against *Internet Worms. Artech House, Inc*., 2004, 319.

[11] **J. Stender, E. Hillebrand, J. Kingdon.** Genetic Algorithms in Optimization, Simulation and modeling. *IOS Press*, 1994.

[12] **K. Ramachandran, B. Sikdar.** Modeling malware propagation in Gnutella type peer-to-peer networks. *Proceedings of Parallel and Distributed Processing Symposium* 2006. *IPDPS*, 2006, *Vol*.20, *No*.25-29, 8.

[13] **L. Corrons.** PandaLabs' Annual Report. *Technical report. PandaLabs*, 2007.

[14] LR Ryšių reguliavimo tarnyba. 2007-ųjų metų tinklų ir informacijos saugumo būklės Lietuvoje tyrimas, įmonių apklausa. *Technical report. LR Ryšių reguliavimo tarnyba,* 2007.

[15] **M. Garetto, W. Gong, D. Towsley.** Modeling Malware Spreading Dynamics. *Proceedings of INFOCOM*, 2003.

[16] **P. Szor.** The Art of Computer Virus Research and Defense. *Addison Wesley Professional*, 2005, 744.

[17] PricewaterhouseCoopers. Information security breaches survey 2006. *Technical report. UK Department of Trade and Industry*, 2006.

[18] **R.R. Hill, G.A. McIntyre, S. Narayanan.** Genetic Algorithms for Model Optimization. *Proceedings of Simulation Technology and Training Conference* (*SimTechT*), 2001.

[19] **S. Staniford, V. Paxson, N. Weaver**. How to 0wn the Internet in Your Spare Time. *Proceedings of the* 11*th USENIX Security Symposium*, 2002, 149-167.

[20] SANS Institute. SANS Top-20 2007 security risks. *Technical report. SANS Institute*, 2007.

[21] The MITRE Corporation. *National Vulnerability Database*, 2007.

[22] TrendMicro Incorporated. Taxonomy of Botnet Threats. *Whitepaper*, 2006.

[23] **Z. Chen, L. Gao, K. Kwiat.** Modeling the Spread of Active Worms. *Proceedings of IEEE INFOCOM* 2003, *IEEE*, 2003, *Vol*.3, 1890-1900.

Received January 2008.