Genetic Algorithms For Tightening Security

Fabrizio Palumbo*, Adam Buji†, Anis Yazidi‡ and Hårek Haugerud§
Artificial Intelligence Lab (AI Lab),
Institutt for informasjonsteknologi, Oslo Metropolitan University, Oslo, Norway
Email: *fabrizio.palumbo@oslomet.no, †adam.buji@hotmaail.no, ‡anis.yazidi@oslomet.no, §hårek.haugerud@oslomet.no

***Abstract*—Proper configuration of operating systems and program parameters is known to be a key security factor in order to remove vulnerabilities. It is known that vulnerabilities can be caused by a human misconfiguration or by an improper chain of parameter settings. It is impossible to find an optimal combination manually due to the enormous number of possible configurations. In this article, we resort to a Genetic Algorithm equipped with a user-defined fitness function in order to compute a configuration of high fitness.**

Our work presents a two-fold contribution. First, we successfully use a GA to implement a moving target defense by alerting the configuration regularly in order to spoil an attacker's reconnaissance efforts. The GA tightens the security solution by evolving the fitness of the configuration over generations while maintaining diversity within generations across a pool of servers. This resulted in high-quality configurations crucial for a successful moving target defense strategy.

Second, we try to find a compromise between tightening the security of the configuration and maintaining the Quality of Service (QoS) on a web server. In practice, usually tightening security on a web server comes at the cost of a decrease in QoS.

I. INTRODUCTION

A. Background

The last two decades of the 21st century experienced an increase in the penetration and expansion of digital technologies. With the decrease in Internet access' costs and information processing, more people are using computers and they expect a quality of service (QoS) compatible with the QoS they are used to getting with other utilities [1].

This service is expected to be available continuously, from anywhere at any time, secure, friendly, and reliable. People expect to log on to the terminal, read mails, make reservations, and do other activities, and with wireless technology, people can access the Internet from anywhere [1].

Web technologies have evolved rapidly in the last five years through web-enabled applications where browsers have become the user interface. Critical functions are done through web applications, such as money transactions, which make web applications attractive targets for hackers [2].

The system can therefore be made more difficult to exploit with better awareness, stronger operating systems, and improved security defense. There is a need for a better understanding of web applications' security since the attackers have moved from attacking the network layer to attacking the application layer [1].

Web applications use the HTTP protocol over the internet by using a web browser which makes it possible to access web applications from anywhere [3]. However, Web applications have bugs in their code that make them vulnerable and they can compromise the system. These vulnerabilities are greater in web applications than in other applications.

More specifically, the operations in a software are controlled by a set of parameters such as the settings of the operating system or the file permissions. These parameters affect both the operating system and the application performance and have therefore implications for the security posture of the system [4]. Some configuration parameters affect the security individually and some do in combination with others.

Many cyber attacks are preceded by a search for vulnerabilities within the network, often caused by a network misconfiguration. Such threats can therefore be eliminated by tuning the correct combination of parameters. Some operating systems offer solutions to prevent security vulnerabilities. However, there is a need for a low-cost method to configure network parameters in order to prevent and defuse cyber attacks.

Genetic Algorithms (GA), a class of heuristic searching algorithms, are applied in this article to discover new, secure, and diverse parameter configurations by modeling a computer configuration as chromosomes, and the individual settings configurations as alleles. The basic idea behind GA is: good chromosomes will generate better chromosomes through a series of selection, crossover, and mutation processes. These processes are stochastic, ensuring therefore a high degree of diversity in the outputs. Additionally, a population of chromosomes can then continuously evolve to become more and more secure in the current environment.

To improve protection against cyber attacks, a continuous change of computer or program parameters configuration can mislead the attacker. The GA will ideally have changed the configurations before the attacker is able to launch his attack, turning obsolete the previously discovered vulnerabilities.

Therefore, GA can significantly improve cyber security by changing the parameters' configurations and increasing their fitness score.

B. Previous Work

A crucial characteristic, that any cyber security defense algorithm needs to posses, is the capability to adapt to new emerging threats and depending on the situation to choose the best reactive action possible. The concept of learning has therefore been applied already in the earliest form of the autonomous defense system, being inspired by reinforcement learning algorithms [5] and from biology itself, taking as

ISBN 978-3-903176-52-2© 2022 IFIP

DOI: http://dx.doi.org/10.23919/WMNC56391.2022.9954297

examples the immune system [6] and the defensive mutualism in microbial symbiosis [7]. However, also cyber attackers can learn and adapt over time [8]. As a consequence traditional static defense techniques (i.e. firewall) are not successful anymore [9]. The moving target (MT) defense approach has therefore emerged and proved successful in increasing network security [9]–[13]. The overall concept of MT is the following: the machine configuration changes as a function of time so that it is more complicated for an attacker to exploit previously identified weak configurations. Importantly, deploying learning algorithms within the MT strategy can improve drastically security by learning over time what parameter configurations are particularly susceptible to cyber attacks [9]–[13]. The idea of deploying evolutionary algorithms in the moving target defense was already introduced in the early 2010s [14]–[17]. GA can play a crucial role in identifying new solutions in the parameters configuration space that would be more and more resilient to cyber-attacks over time. The Works of Smith [18] and Zhou [19] show that an evolutionary strategy is a powerful approach when defending against cyber-attacks. Smith first investigates the application of GA to learn secure parameter configurations, also called chromosomes, and to implement the moving target defense (MT) strategy. A computer's genetic code, or DNA, includes all the settings of all of its parameters. GA then evolves, over iterations/generations, more secure configurations which are then used to immunize the machine to attacks. In his work, Smith [18] also implements a beam search-based system to select which chromosomes are inherited across generations. This approach outperformed GA in increasing average configuration fitness while still maintaining high diversity. Experimentation showed that a prototype system using these strategies, for a small number of attacks per generation, was often successful in creating a new generation of chromosomes that were immune to attacks from the previous generation [18] [19]. Additionally, it is also introduced the idea that the efficiency of GA is improved by using machine learning to classify generated solutions and filtering the sub-optimal low fitness configurations [18] [19].

Software configuration consists of parameters and through them, it is possible to control aspects of the system. Therefore, a misconfigured software, by a single parameter or a combination of parameters, exposes the system to vulnerabilities. The huge number of parameters makes it hard to identify vulnerable settings that can be attacked. Moreover, combinations of settings might cause hidden vulnerabilities, complicating the problem.

A first solution to the problem exploits the identification of common features across vulnerable configurations. In her Ph.D. thesis, Dr.Oddell [20] developed a method to detect security-relevant parameters. By analyzing configurations that are vulnerable to the same exploit and comparing the similarity of their parameters it is possible to highlight the ones that are vulnerable to a specific attack. Genetic Algorithms are then used to generate the configurations used for the identification of vulnerable parameters.

Another approach is based on the time of the incorrect function of a system, which can be caused by errors in its configuration. In [21], the authors address the problem of diagnosing configuration errors. For example, changing the local firewall policy could cause a network-based application to malfunction. This approach searches the time point at which the system transitioned into malfunctioning. The cause of the failure is then identified by comparing the system configuration before and after the malfunction. Whitaker et al [21] implement a tool called Chronus to reduce the need for human expertise. Chronus automates the search for failure-inducing state changes and identifies the configuration errors. However, this tool requires, among other utils, user-written software to find out if the system is currently working and only then finds the error.

A final approach is presented by Zhang et al. [22] and it is implemented in the tool called EnCore. EnCore detect software misconfiguration by taking into account the correlations between configuration entries and their interaction with the executing environment. Encore consists of four steps; data collecting, data assembling, rule generator, and anomaly detection. It is able to learn a broad set of configuration anomalies that span the entire system and detect real-world problems as well as injected errors.

The work presented in this article builds on previous research and aims to improve on its weaknesses as follow:

- By using real parameters for each configuration, allowing us to replicate possible machine misconfigurations leading to attack vulnerability.
- By directly simulate attackers targeting the system, allowing us to directly test the security level.

II. Methods

This article uses Apache v2.2 configuration parameters as defined in STIG [23], which stands for Security Technical Implementation Guides. The STIG was developed by The Defense Information Systems Agency in order to implement configuration guidelines for systems that are deployed across the Department of Defense. The scores are based on the CVSS scoring system [24], according to the results of the test code. In this article, a part of the Apache v2.2 configuration was used as proof of the GA concept and its capability of finding the fittest solution.

DiSA's Security Technical Implementation Guides gather all configuration parameters that contribute to known vulnerability attack paths.

STIG provides a network administrator with an explanation on how these configurations can contribute to a vulnerability and how it can be fixed [23].

A. Genetic algorithm server

The chromosomes represent a configuration's combination of parameters. The size of each chromosome is based on the configurations' parameters, where one allele or more represent a parameter. In this research, the chromosome size is 25 and

the total number of chromosomes in one generation is 40, which is enough to keep the parameters' diversity according to the number of parameters. The number of solutions in one generation has been determined after many experiments of the algorithm. It is possible to increase the number of chromosomes in one generation; this might increase the diversity, but at the same time it would cost more computational resources.

All parameters are represented as a binary presentation, so the parameter that needs to be integers is converted to integers while converting the chromosomes into configurations. The parameters represented as "element from a list" are also converted from binary to element from a list while converting the chromosome into a configuration.

Two algorithms have been developed and a total of four scripts have been used.

- The genetic algorithm will generate security solutions.
- The fitness score algorithm is responsible for providing the genetic algorithm with the fitness scores of the security solutions. The scoring system relies on previous preferences from STIG which provides the vulnerabilities, the parameters responsible for them, and the solutions.

In the beginning, the first chromosomes are initiated randomly using a uniform distribution, and then saved to the population's pool.

In a Multi-point crossover, two or more random values are selected and at these selected points, the variables are exchanged between the individuals as shown in Figure 1.

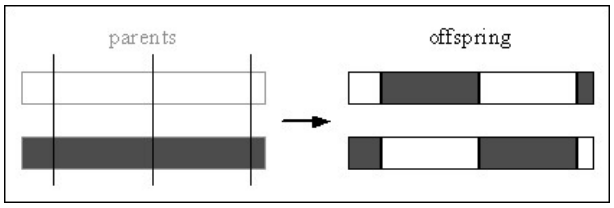

Fig. 1. multi-points crossover

The experiments presented in Figure 2 and 3 show that the multi-point crossover is more effective than the single-point crossover because it increases the probability of the new chromosomes getting the fit parts of both parents [25].

If we consider the following two individuals with 10 binary variables in each and the selected points are(4,7):

chromosome 1	0 1 1 0 ‖010‖011
chromosome 2	0 0 1 0 ‖100‖101

The generated new chromosomes would be:

chromosome 3	0 1 1 0 ‖100‖011
chromosome 4	0 0 1 0 ‖010‖101

After executing the crossover function it is necessary to run a mutation process in order to prevent the algorithm from being trapped in a local minimum. The mutation is an operator which maintains genetic diversity within the population. The mutation process introduces a new genetic structure into the population by modifying some of the alleles. [25].

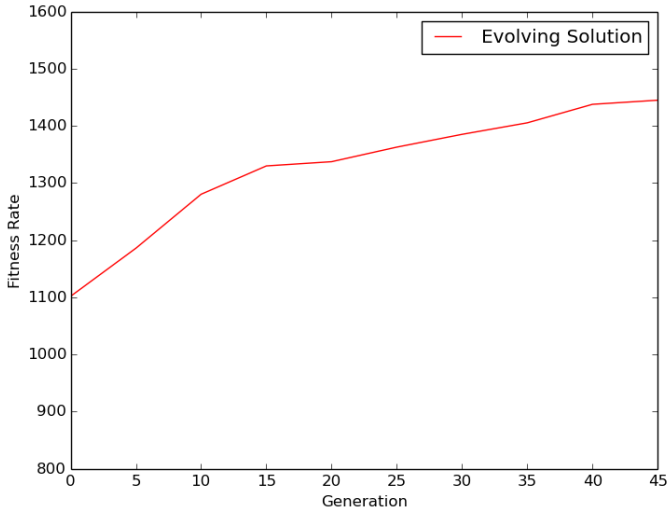

Fig. 2. Single crossover

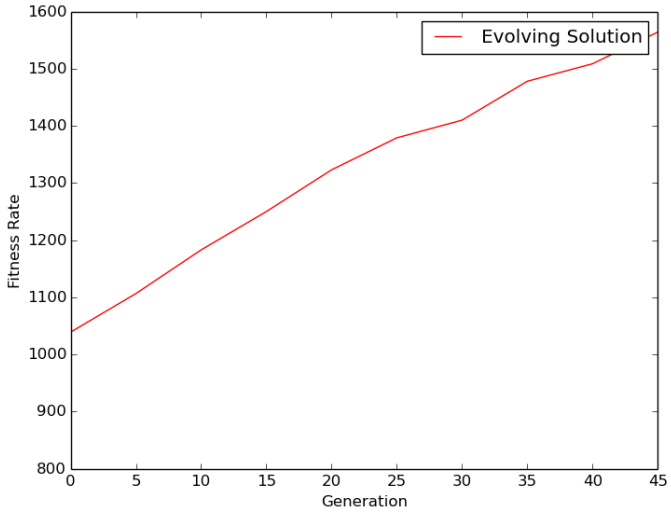

Fig. 3. Multi-point crossover

In this article, binary representatives were used, so the mutation process was done by flipping a random allele in the chromosome to the opposite value as shown in the following example:

If we consider the following two individuals with 10 binary variables in each and the selected allele location is 5:

chromosome 1	0 1 1 0 —0 — 1 0 0 1 1

In the mutation process, it would flip the value from 0 to 1, so the generated new chromosome would be:

chromosome 2	0 1 1 0 —1 — 1 0 0 1 1

After executing the crossover and mutation process, the algorithm divides the existing solutions in the population's

pool between the available docker containers in the farm. This is done to compute the fitness scores and select the fittest solutions of the whole population. We then select the best n solutions for further mating and to produce the next generation.

When the algorithm sends the solutions to the web servers, it makes sure it sends them only to available web servers. It checks their availability before sending the solutions in order to avoid any future errors, and ensure that all the solutions are tested as planned. After sending the solutions to VM's farm, the algorithm opens a port temporarily to receive the scores for each solution. After receiving all the scores, the algorithm selects the n fittest solutions for further mating.

The algorithm runs 40 generations and the experiments show that 40 generations are good enough to reach a solution close to the optimal fitness. However, the generation number can be increased in case of the need to improve a more complex chromosome to optimize the solution.

B. Reconfiguring the Apache server

The Genetic Algorithm generates solutions and sends them to the VMs farm which then applies those configurations to the Apache server in order to execute the automated attacks.

The VM receives the solutions as a list and through an agent script, it translates the list into configurations and then applies them to the Apache server.

C. Quality of service

It is important to make sure that a good QoS is maintained while tightening the security.

The QoS was measured by the total used time for the following processes: Lookup time, Connect time, Pre-transfer time, and Start-transfer time. The total used time was measured by running the "curl" command while pressing the web server using "httperf" on two web servers; one web server with the fittest configuration, and another with a vulnerable configuration. The experiments on both web servers were done under the same conditions which contain six load stress levels. The experiment was run 10,000 tests/webpage requests on each web server on every stress load level which makes it 60,000 tests/webpage requests in total on each web server, where each level had a different load rate generated by httperf tool, and is this explained in more detail later.

III. Results

A. Security solutions

In this article we show experimentally, by using real parameter configurations and simulated cyber-attacks, that evolutionary algorithms represent a powerful tool to improve cyber security.

The experiments were conducted using configurations of 24 Apache 2.2 parameters.

We show that by using GA it is possible to improve system security over the course of generations. Through iterations of trial and error, we identify that the best performance was achieved by:

- using a population size consisting of 50 individuals.
- running for 40-45 generations.
- One or two genes are modified in every mutation, using elitism as the deterministic tournament selection, and using a two-point crossover.

The start point solution was initialized randomly. In the documented experiment, the initialized solution was 1035.2 when using a one-point crossover, and it was 1011.6 when using a two-point crossover. The fittest solution was 1490.0 when using a one-point crossover and 1584.2 when using a two-point crossover as shown in Figure 4.

The experiment shows maintenance of diversity which is required in order to ensure the solution space is checked thoroughly enough, especially in the earlier stages of the process. Population diversity is considered to be the main reason for premature convergence.

The experiment also shows that the diversity was high in the start and it became lower and lower in the later stages, as shown in Figure 5. The results also show that crossover types significantly impacted fitness or diversity in this article, as shown in Figure 2 and Figure 3.

A high diversity in the population generates a different solution from generation to generation implementing a moving target defense. This results in wasting the attacker's reconnaissance effort by changing the configurations continuously until reaching the fittest solution possible.

To quantify the impact of our approach on the QoS for the user, we focus on the total time used between establishing a connection and data transfer, as shown in 6. The mean value of the total time of all test levels for the web server with the fittest configuration was 0.004062783, while for the web server with the vulnerable configuration was 0.002006883. Therefore, using the fittest configuration solution increase the time necessary to send the requested page, compared to the vulnerable solution.

B. Fitness score

This article investigates the role of the GA in improving cyber security. We define a fitness value for each chromosome, as an estimate for security. Those values were then used to select configurations with higher fitness values.

The first scores were decided according to randomly generated chromosomes, which varied from one experiment to another. Then, the GA performed the selection based on those scores. Crossovers also used the same scores to decide which chromosome moves forward, which are based on random selection and affect the evolution process.

The mutation process took place after the crossover with randomly selected candidates preventing the algorithm to fall into local minima.

The scoring system was built over the course of three experiments:

- An offline scoring system based on information from the STIG database.
- An online-scoring system based on the OWASP vulnerability scanning

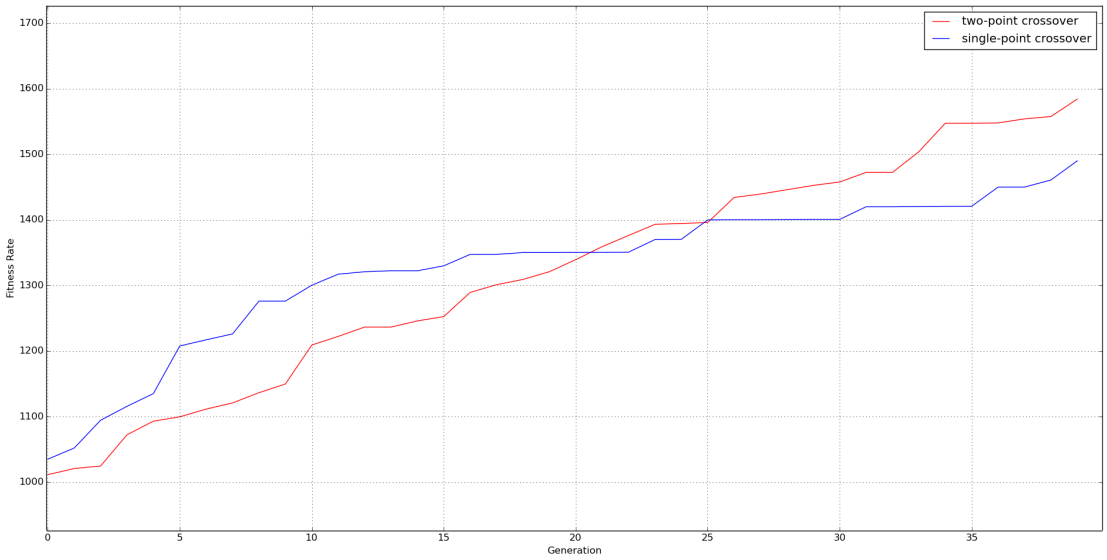

Fig. 4. Security solution's evolving

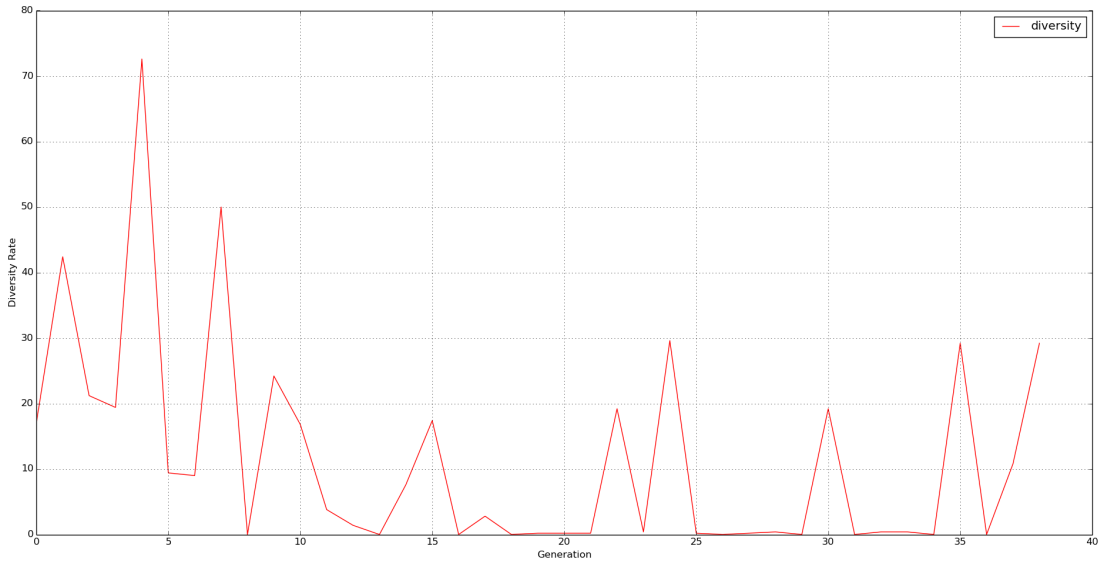

Fig. 5. Diversity

- Real-life attacks were executed on both the most vulnerable solution and the fittest solution

The GA successfully generated a more secure configuration that was not vulnerable to real-life attacks.

The QoS is an important factor to consider while improving the security solution. The conducted experiment shows that tightening the security does affect the QoS.

The experiment reported in Figure 6 and Table 1 shows a significantly longer time required for secure configuration to start transferring once a connection is established. The average time used by the fittest configuration was 0.004062783, while the average time used by the vulnerable configuration was 0.002006883 which is a significant difference.

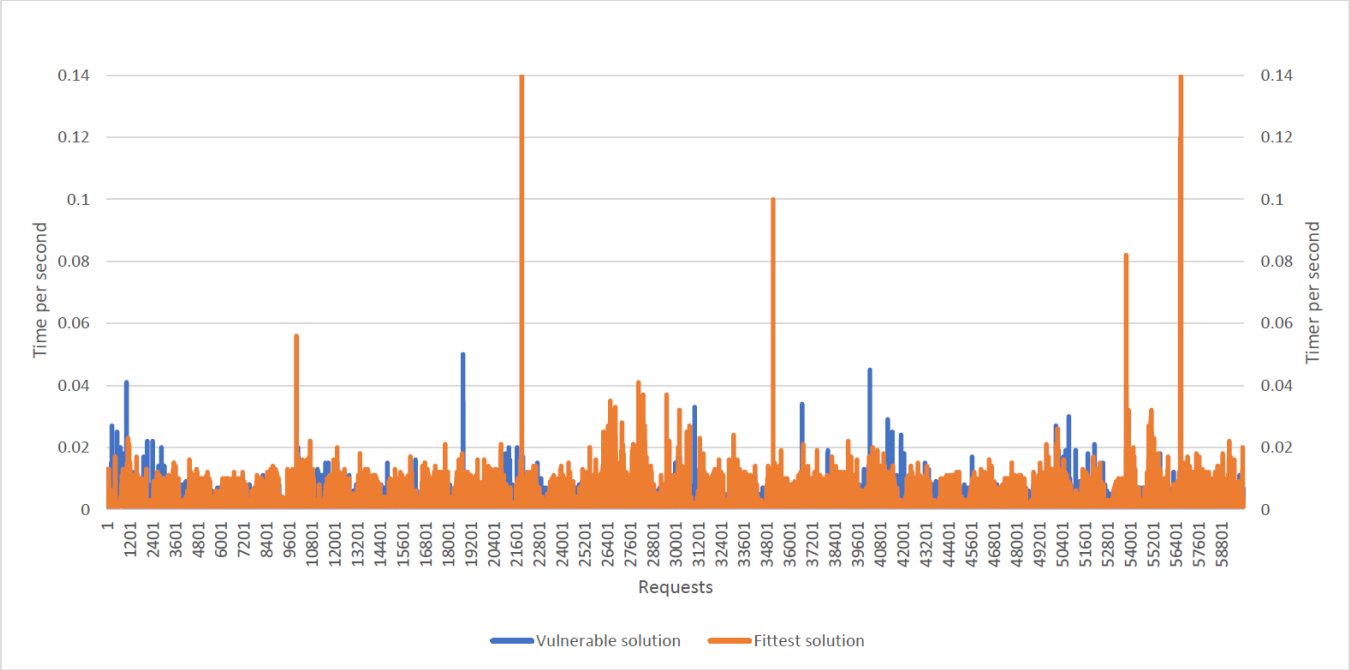

Fig. 6. Security impact on QoS of webserver

	Fittest solution	Vulnerable solution
Mean	0.004062783	0.002006883
Standard Deviation	0.004011096	0.001806554
Total tests	60000	60000
Variance	1.60889E-05	3.26364E-06

Table 1. Security impact on QoS of webserver

More investigation and experiments are required in the QoS area to make a concrete analysis. Time constraints made it impossible to conduct a more thorough investigation in this article.
The total fitness (100%) in this article was defined as:

- the configuration fitness (70%)
- Moving target success (10%)
- the 2 attacks experiments' success (10%)
- the QoS fitness (10%)

The Configuration's fitness was calculated according to the expected highest fitness (1700), and the fittest solution was 1584.2. The moving target was calculated according to the change in every five generations: a change in the configuration in every five generations is considered a finding. The 2attacks score was defined by the success or failure of the two cyber attack strategies tested: Denial of service and buffer overflow. The QoS fitness was calculated according to how many of the 60000 tests still got a comparable response time between the case of the fittest security solution and the case of a vulnerable solution (<0.006s), which was 45620 in the conducted experiments.

$$f(TFitness) = \sum ConfScore + \sum MovingTargetScore + \sum 2AttacksScore + \sum QoSScore \quad (1)$$

$$f(TFitness) = 65,2 + 10 + 10 + 7.6 \quad (2)$$

$$TotalFitness = 92.8\% \quad (3)$$

IV. Discussion

This article shows that GA can contribute to reducing system vulnerabilities by detecting and solving misconfiguration. The GA evolves fitter solutions within every iteration, reducing vulnerabilities over generations.

Vulnerabilities caused by human misconfiguration are hard to find manually, due to the huge number of parameters, and it is even harder to find the right chain of parameters. However, GA can automate this process and provide more and more accurate solutions over the course of generations.

The implemented algorithm aims to find the fittest configuration in Apache 2.2 server, fixing the vulnerabilities caused by human misconfiguration. This algorithm can then replace manual human work which is impossible to consider due to time constrain.

The GA manages to reach the fittest solution possible through 40 generations, finding the correct parameters and the correct chain of parameters.

The GA preserves diversity within each generation and maintain a diverse solution from generation to generation. This is particularly important since it can mislead an attacker that has done reconnaissance between the generations.

A. Fitness

The main objective of this article is to reach the fittest solution through the use of GA. The fitness function showed flexibility in a constantly changing environment. The fitness of the chromosome is based on a discovered security concern that has been run through the penetration test. The test sends back the security level of each chromosome with the goal to reach as few vulnerabilities as possible.

The CVSS score of a parameter setting, which has been used to score the security, is based on the effect it might cause on information security such as confidentiality, integrity, and availability.

The scoring system starts with simulated attacks, based on information from STIG. In the second step, the scoring system is developed to run a vulnerability scanning using OWASP, showing results close to the simulated attacks. Finally, the security level is tested through real-life attacks, showing results as expected.

B. Moving target

Implementing a moving target defense strategy is an objective of this article. It protects from attacks by changing parameter configurations continuously, so in case the attacker did his reconnaissance, it would be non-effective because of the change in the configuration. Diversity is therefore an important factor during the evolution process to implement a moving target defense within each generation.

Keeping the diversity of high-quality configurations is important within the moving target strategy since different configurations are applied on a number of machines.

The difference between the fitness of two different solutions in one generation represents the diversity within a generation. The diversity from generation to generation is calculated by looking at the differences between the fittest solution in the generation and the following generation.

In this article, the diversity is maintained across generations which implements a moving target defense until the fittest solution possible is reached. This misleads an attacker in case of configuration reconnaissance. The diversity also prevented the algorithm from falling into a local minimum, which helped it to evolve the fittest solution.

C. Quality of Service

Improving security is an important factor, but at the same time, it should be parallel with having good QoS. In this article, the QoS was investigated in order to see if there is any noticeable relation between security and QoS. The conducted experiments do show a negative impact on the QoS when improving security, in terms of total used time between when a connection was established and when the data actually began to be transferred. The impact was measured while the web server was under continuous stress of HTTP requests load. The average used time, from starting the connection to starting the data transfer, significantly increase for server configurations with tighter security.

V. Conclusion

The objective of this article is to improve the security solutions by applying the genetic algorithm to the configuration parameters of Apache2.2. The genetic algorithm is applied successfully allowing the configurations to evolve to be diverse and more secure over generations. Additionally, this approach also allows for a moving target defense by changing the configuration from generation to generation until reaching the fittest solution possible.

Vulnerabilities can be caused by a misconfiguration or by an unlucky chain of configurations. This is difficult or even impossible for the system administrator to discover manually due to the huge amount of parameters, and big amount of possible combinations. Therefore, a Genetic Algorithm was used to find more secure configurations. Configurations were represented as chromosomes and the GA took those through a series of selection, crossover, and mutation processes which resulted in more secure configurations across each generation.

The results demonstrate the performance of the evolutionary approach for managing configurations consisting of 24 parameters from Apache 2.2. The simulated attacks of these configurations are based on information from the STIG database. The genetic algorithm discovers better parameter settings for the attacked parameters in each generation.

At the first stages, the solution fitness improves significantly. A reasonable level of diversity is maintained. It starts with a high level of diversity because the parameters are randomly initialized. In the late stages, the fitness improvement decrease alongside a decrease in diversity.

The experiment showed that the diversity within the generation maintained the ability to have a diverse configuration from generation to generation. Changing the configuration from generation to generation creates a moving target that misleads an attacker based on reconnaissance.

In terms of security, this experiment demonstrates resilience.

As an added **contribution** of this paper, the genetic algorithm manages to find the fittest solution possible, which helps prevent attacks caused by a misconfiguration or by a poor chain of parameters. This is considered to be a new concept in improving security.

As an added **contribution** of this paper, the genetic algorithm helps prevent attacks by spoiling an attacker‘s reconnaissance efforts by continuously changing the configuration. A series of changes in the configuration from generation to generation makes it possible to enhance security by deploying a moving target defense.

As an added **contribution** of this paper, the investigation of the relationship between security and QoS shows that the security level has a significant impact on the QoS. Improved security results in a longer time delay between establishing the connection and the start of data transfer. This might be

a reasonable price to pay for having a higher security level. However, to be more concrete about the impact of this delay on the QoS, it is recommended that further investigation -as future work- covering other areas in QoS are conducted.

REFERENCES

[1] Y. Liao, V. Vemuri, Enhancing computer security with smart technology (2006).

[2] A. Tsalgatidou, T. Pilioura, An overview of standards and related technology in web services, Distributed and Parallel Databases 12 (2-3) (2002) 135–162.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext transfer protocol–http/1.1, Tech. rep. (1999).

[4] J. Oberheide, E. Cooke, F. Jahanian, If it ain't broke, don't fix it: Challenges and new directions for inferring the impact of software patches., in: HotOS, 2009.

[5] L. Beaudoin, Autonomic computer network defence using risk states and reinforcement learning, 2009.

[6] S. Hofmeyr, S. Forrest, Architecture for an artificial immune system, Evolutionary computation 8 (2000) 443–73. doi:10.1162/106365600568257.

[7] S. Stolfo, Symbiotes and defensive Mutualism: Moving Target Defense, 2011, pp. 99–108. doi:10.1007/978-1-4614-0977-9_5.

[8] M. L. Winterrose, K. M. Carter, Strategic evolution of adversaries against temporal platform diversity active cyber defenses, ADS '14, Society for Computer Simulation International, San Diego, CA, USA, 2014.

[9] E. W. Fulp, H. D. Gage, D. J. John, M. R. McNiece, W. H. Turkett, X. Zhou, An evolutionary strategy for resilient cyber defense, in: 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1–6. doi:10.1109/GLOCOM.2015.7417814.

[10] M. Carvalho, R. Ford, Moving-target defenses for computer networks, IEEE Security Privacy 12 (2) (2014) 73–76. doi:10.1109/MSP.2014.30.

[11] P. Pal, R. Schantz, A. Paulos, B. Benyo, D. Johnson, M. Hibler, E. Eide, A3: An environment for self-adaptive diagnosis and immunization of novel attacks, in: 2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2012, pp. 15–22. doi:10.1109/SASOW.2012.13.

[12] P. Pal, R. Schantz, A. Paulos, B. Benyo, Managed execution environment as a moving-target defense infrastructure, IEEE Security Privacy 12 (2) (2014) 51–59. doi:10.1109/MSP.2013.133.

[13] D. J. Musliner, J. M. Rye, D. Thomsen, D. D. McDonald, M. H. Burstein, P. Robertson, Fuzzbuster: Towards adaptive immunity from cyber threats, in: 2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2011, pp. 137–140. doi:10.1109/SASOW.2011.26.

[14] M. Crouse, E. W. Fulp, A moving target environment for computer configurations using genetic algorithms, in: Configuration Analytics and Automation (SAFECONFIG), 2011 4th Symposium on, IEEE, 2011, pp. 1–7.

[15] M. Crouse, E. W. Fulp, D. Canas, Improving the diversity defense of genetic algorithm-based moving target approaches, in: Proceedings of the National Symposium on Moving Target Research, 2012.

[16] D. J. John, R. W. Smith, W. H. Turkett, D. A. Cañas, E. W. Fulp, Evolutionary based moving target cyber defense, in: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO Comp '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 1261–1268. doi:10.1145/2598394.2605437.
URL https://doi.org/10.1145/2598394.2605437

[17] D. Zegzhda, D. Lavrova, E. Pavlenko, A. Shtyrkina, Cyber attack prevention based on evolutionary cybernetics approach, Symmetry 12 (11). doi:10.3390/sym12111931.
URL https://www.mdpi.com/2073-8994/12/11/1931

[18] R. W. Smith, Evolutionary strategies for secure moving target configuration discovery, Ph.D. thesis, Wake Forest University (2014).

[19] X. Zhou, Measurements associated with learning more secure computer configuration parameters, Ph.D. thesis, Wake Forest University (2015).

[20] C. A. Odell, Using genetic algorithms to detect security related software parameter chains, Ph.D. thesis, Wake Forest University (2016).

[21] A. Whitaker, R. S. Cox, S. D. Gribble, Configuration debugging as search: Finding the needle in the haystack., in: OSDI, Vol. 4, 2004, pp. 6–6.

[22] J. Zhang, L. Renganarayana, X. Zhang, N. Ge, V. Bala, T. Xu, Y. Zhou, Encore: Exploiting system environment and correlation information for misconfiguration detection, ACM SIGPLAN Notices 49 (4) (2014) 687–700.

[23] [Accessed Mars. 08 2017] (Mars 2017). [link].
URL https://www.stigviewer.com/

[24] [link].
URL https://www.first.org/cvss/specification-document

[25] S. Sivanandam, S. Deepa, Introduction to genetic algorithms, Springer Science & Business Media, 2007.