ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20250218002392
Original scientific paper

Advanced Security in Distributed Web Systems Using Genetic Algorithm-Based Techniques

Doru Anastasiu POPESCU*, Marian ILEANA, Nicolae BOLD

Abstract: The growing sophistication of cyber threats has heightened the need for enhanced security measures in distributed web systems. This paper explores the potential of genetic algorithms to reinforce these systems by improving their adaptability to emerging vulnerabilities. Genetic algorithms, rooted in evolutionary principles, provide a mechanism to optimize security parameters, such as threat detection, intrusion response, and anomaly identification. Unlike traditional methods, these algorithms offer dynamic solutions capable of evolving with changing attack patterns. The study delves into various ways genetic algorithms can be incorporated into security frameworks for distributed systems, focusing on how these adaptive strategies could increase system resilience and robustness. Key considerations include the efficiency of configuration optimization, the ability to manage diverse security scenarios, and the scalability of algorithmic integration across multiple nodes within a system. By analyzing theoretical models and current advancements in evolutionary computing, we provide a comprehensive evaluation of the algorithm's potential impact on distributed security infrastructures. This research highlights the importance of innovative, adaptive approaches to maintaining the security of increasingly interconnected web systems in the face of evolving cyber risks.

Keywords: cybersecurity; distributed web systems; evolutionary techniques; genetic algorithms; network protection; security optimization

1 INTRODUCTION

In today's digital age, the security of distributed web systems is a major challenge due to their inherent complexity and heterogeneity [1]. If resources and services are distributed across multiple nodes, each connection point can be vulnerability. Therefore, creating effective and strong security methods is vital to protecting data and ensuring the integrity of computer systems [2].

Mathematical modelling offers a rigorous framework for analyzing and designing security solutions. By simulating attack vectors and system responses, researchers can quantitatively evaluate the effectiveness of security methods and iteratively optimize their performance using computational techniques [3]. Due to their ability to explore large solution spaces and find optimizations that may be inaccessible to traditional methods, genetic algorithms have proven to be useful tools in this context [4].

This study focuses on the integration of mathematical modelling with genetic algorithms to develop advanced security techniques in distributed web systems [5]. The main goal is to improve protection against cyber threats by identifying and implementing effective solutions [6].

With the increase in the number of critical web applications, cyber threats have diversified and intensified. Distributed Denial of Service (DDoS) attacks, unauthorized access, interception of sensitive data, and manipulation of communications are just a few examples of risks affecting distributed systems [7]. In this context, organizations must implement proactive security measures that are flexible enough to respond quickly to emerging attacks. This requires advanced monitoring technologies and adaptive defense methods [8].

Automation of security processes plays a crucial role in detecting and preventing cyberattacks [9]. Artificial intelligence and machine learning algorithms are capable of analyzing large volumes of data in real time, identifying anomalies or suspicious behaviors. Genetic algorithms, in particular, can generate and adapt defense solutions iteratively, providing a level of continuous optimization that cannot be achieved by traditional methods [10].

Genetic algorithms were originally developed as a method for solving complex problems inspired by the principles of natural evolution, such as selection, mutation, and genetic recombination [11]. In recent decades, they have been successfully applied in various fields, including computer security. The major advantage of genetic algorithms lies in their ability to explore a large solution space and discover efficient combinations that are not analytically obvious. Mathematical models provide a framework to simulate attack scenarios, while GAs iteratively optimize defense parameters based on these simulations [12].

Adopting international standards, such as ISO/IEC 27001 and the NIST Cybersecurity Framework, can support the development of more secure distributed architectures. Integrating these standards into the proposed methodology contributes to standardizing security measures and increasing user trust in digital infrastructures [13].

Traditional security approaches, such as static firewalls and rule-based intrusion detection systems, struggle with complex attack vectors due to their inability to adapt to new threat patterns. These systems often fail to address zero-day exploits, dynamically evolving attack surfaces, and the heterogeneity of distributed nodes, leading to delayed threat response and increased vulnerability gaps.

The fundamental principles of genetic algorithms are discussed in detail, as well as how they can be used to design and optimize security techniques [14]. The proposed approach, based on compound chaotic hash functions, provides a robust theoretical framework and practical tools for improving the security and resilience of distributed web systems, with relevant applications in areas such as blockchain and data authentication [15]. Thus, this work not only brings novel elements but also provides a basis for future research in the optimization of distributed systems through mathematical and algorithmic techniques.

2 LITERATURE REVIEW

2.1 Key Contributions to the Field

Research carried out by Abbas et al. [16] presents a secure method of routing within IoT (Internet of Things) networks by integrating blockchain for authentication and using genetic algorithms for routing optimization. Authentication based on blockchain technology guarantees the integrity and security of the credentials of IoT nodes, protecting them against tampering. Genetic algorithms used by SDN controllers optimize routes to reduce the energy consumption of nodes. In addition, the proposed systems can detect and limit malicious nodes to prevent network compromise. Simulations show that this technique significantly improves network efficiency, which means less resource consumption and fewer lost packets. This makes IoT networks more secure and efficient. The integration of blockchain for authentication and genetic algorithms for routing optimization provides an efficient and secure solution for IoT networks, ensuring integrity, security, and operational efficiency.

The study by Tahir et al. [17] uses an innovative security model that protects cloud data using Genetic Algorithms (GA). Using GA for cryptographic key generation, the proposed Crypto GA model ensures the encryption and decryption of data to maintain its integrity and confidentiality. The system creates 128 bit keys and uses genetic operations such as crossover and mutation to obtain ciphertext. To evaluate the performance of Crypto GA, tests were performed on ten different data sets and compared with state-of-the-art cryptographic algorithms such as DES, 3DES, RSA, Blowfish, and AES. Experiments have shown that Crypto GA is powerful and efficient, providing higher execution times, throughput, key sizes, and avalanche effects than competing methods. The GA-based Crypto GA model is a robust solution for securing data in the cloud, outperforming established cryptographic algorithms in terms of integrity, privacy, and operational efficiency.

A study by Patil and Bansode [18] proposes an innovative feature selection method for detecting attacks on web servers. By combining mutual information (MI) and genetic algorithm (GA), the research aims to improve the accuracy and reduce the computational cost of machine learning classifiers. The proposed approach reduces the set of relevant features to only four, improving the accuracy of classifiers such as Naive Bayes, k-Nearest Neighbors, Random Forest, and SVM. The results show accuracy increases of up to 13.11% and a 99% reduction in computational time compared to existing methods. This study highlights the importance of effective feature selection in the context of cyberattacks by providing a hybrid method that combines the advantages of filtering and wrapper-based learning techniques.

2.2 Examination of Authors and Research Teams

A thorough examination of various research papers on "Genetic Algorithms and Security" was carried out through a dynamic analytical approach. The study utilized the Dimensions.ai platform as both a data source and search engine. For the data analysis process, VOS Viewer software was employed to identify leading contributors in the field [19, 20]. The results were presented using a graphical network representation. The dataset comprised 10718 authors, with a relevance threshold set at a minimum of five publications and at least five citations per author. Based on these criteria, 219 authors qualified. The largest interconnected network of authors consisted of 204 members (refer to Fig. 1).

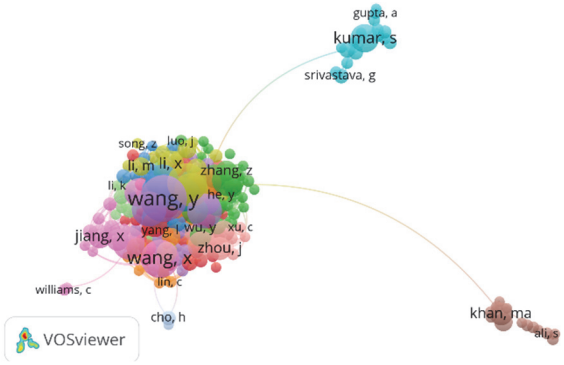

Figure 1 An author network map on "Genetic Algorithms and Security" highlights leading researchers and collaboration patterns

A map illustrating the geographical distribution of research units based on author affiliations was created using the same dataset. Countries needed at least five publications with a minimum of five citations each to qualify. Out of 129 countries, 77 met the criteria. This visualization highlights global research hubs in the field (see Fig. 2).

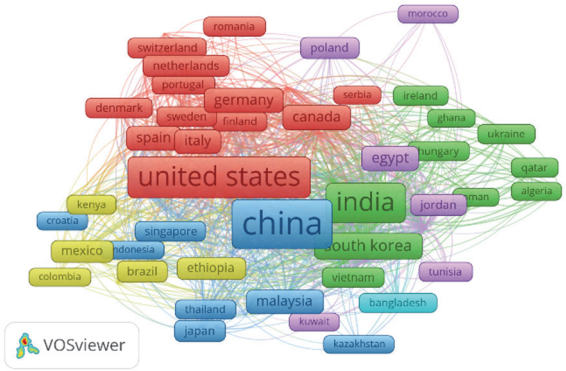

Figure 2 A country-based map for "Genetic Algorithms and Security" illustrates the geographical distribution of research contributions

3 METHODOLOGIES

3.1 Distributed Web Systems

Distributed web systems consist of interconnected nodes that collaborate to provide services and share resources across multiple physical locations [21]. These systems are designed to improve scalability, reliability, and performance by distributing workloads across different servers and networks. However, the distributed nature introduces complexities related to data synchronization, fault tolerance, and, most importantly, security [22].

The architecture of a distributed web system typically includes multiple components, such as application servers, database nodes, load balancers, and client access points, which must work together in a coherent manner [23]. Each of these components represents a potential attack surface, making security measures essential in system design. The dynamic interaction between nodes requires continuous

monitoring and adaptive response mechanisms to mitigate evolving cyber risks [24].

One of the main challenges in distributed systems is maintaining consistency and availability while ensuring data integrity and confidentiality [25]. Traditional security approaches, such as static firewalls and rule-based intrusion detection systems, often fail to cope with the complex attack vectors that target these systems. Therefore, there is a growing need for more adaptable security frameworks that can respond to threats in real time [2].

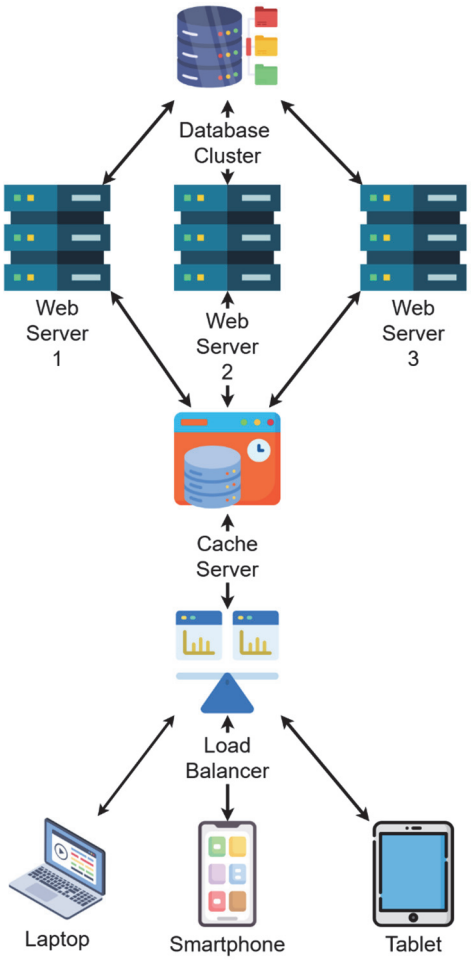

Figure 3 Architecture of a Distributed Web System for load balancing, caching, and redundancy optimization

The diagram in Fig. 3 illustrates the architecture of a distributed web system, focusing on scalability, performance, and fault tolerance. This system design ensures efficient load distribution, faster data retrieval through caching, and redundancy at multiple layers for high availability. The components that make up this distributed system architecture are:

1. Devices (Clients):
   - Laptop, Smartphone, and Tablet represent user devices that make requests to the web system.
2. Load Balancer:
   - Distributes incoming client requests across multiple web servers to ensure efficient resource use, reduce latency, and prevent overload on any single server.
3. Cache Server:
   - Positioned between the web servers and the load balancer. It stores frequently accessed data to reduce database queries and improve response times.
4. Web Servers:
   - Three web servers (labelled Web Server 1, Web Server 2, and Web Server 3) handle client requests. They provide redundancy and scalability.
5. Database Cluster:
   - Serves as the central storage system. It aggregates and synchronizes data across multiple nodes to enhance reliability, ensuring data consistency and availability.

In this context, genetic algorithm-based techniques offer significant advantages. These algorithms are able to evolve and optimize security configurations dynamically, using the principles of natural selection. For distributed web systems, genetic algorithms can optimize parameters related to encryption protocols, intrusion detection thresholds, and resource allocation strategies, thus improving both performance and security [26].

This section serves as a foundation for the methods discussed in this paper, providing a summary of distributed web systems, their inherent security vulnerabilities, and the need for adaptive and algorithmic solutions. The following sections will discuss various ways in which genetic algorithms can be modified to successfully address these issues.

3.2 Genetic Algorithms

Evolutionary computing algorithms are based on Darwin's theory of evolution. In his 1960 paper 'Evolutionary Strategies', I. Rechenberg introduced the concept of evolutionary computation [27].

In our implementation, mutation probability p_m was set to 0.01 to balance exploration and exploitation, while crossover probability was fixed at 0.8 to encourage diversity. These values were empirically determined through preliminary simulations.

Optimization methods based on natural evolution are known as genetic algorithms (GA). To find the best solutions to complicated problems, they use chromosomes, a population of candidate solutions [28].

Initialize the population:

$$P(0) = \{x_1, x_2, ..., x_N\} \tag{1}$$

where N is the initial population size.
Fitness assessment:

$$f(x_1), \text{ for } i = 1, 2, ..., N \tag{2}$$

Selection: The selection probability of a chromosome x_i can be proportional to its fitness:

$$p_i = \frac{f(x_1)}{\sum_{j=1}^{N}(x_j)} \tag{3}$$

Crossover: Generation of offspring by combining two parents. For example, with a simple crossover at one point k:

$$\text{If } x_i = \left(x_{i1}, x_{i2}, ..., x_{im}\right) \text{ and } x_j = \left(x_{j1}, x_{j2}, ..., x_{jm}\right) \quad (4)$$

$$Downward\ y = \left(x_{i1}, x_{i2}, ..., x_{ik}, x_{j(k+1)}, x_{j(k+2)}, ..., x_{jm}\right) \quad (5)$$

Mutation: Random gene modification. For a binary gene:

$$x_{ij} = \begin{cases} 1 - x_{ij}, \text{ with probability } p_m, \\ x_{ij}, \text{ else.} \end{cases} \quad (6)$$

Here, p_m is mutation probability, the chance of a gene mutating.

Update the population:

$$P(t+1) = \{y_1, y_2, ..., y_N\} \quad (7)$$

where y_i are the descendants of generation t.

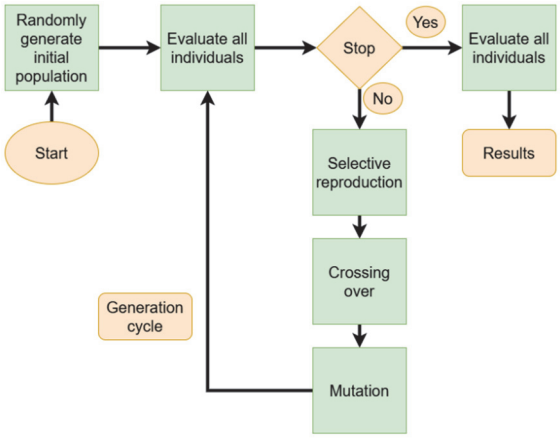

Figure 4 Execution flow diagram of a genetic algorithm for population optimization

Fig. 4 illustrates the complete process of a genetic algorithm, used for optimizing solutions. The process begins with the random generation of an initial population. Each individual in this population is evaluated according to a fitness function. If the stopping conditions are not met, the algorithm goes through the stages of selective breeding, crossover, and mutation to create a new generation. Then, the process is repeated in successive cycles until the stopping conditions are reached, after which the final results of the evaluation of the optimized population are displayed.

3.3 Optimization of Encryption Protocols with Genetic Algorithms

Advanced encryption algorithms protect private and clean data during transit and storage [29]. The architecture of these protocols must be adaptable to distributed web systems to respond to increasingly complex cyber threats and attacks. Genetic algorithms (GAs) are a promising method for optimizing encryption protocols that generate stable and dynamic solutions by using natural selection techniques [30].

Genetic algorithms are capable of generating cryptographic keys of varying complexity through the iterative process of selection, crossover, and mutation [31]. This process involves:

- Population initialization: An initial set of candidate keys (chromosomes) is generated, each representing a binary sequence usable in encryption.
- Fitness evaluation: A fitness function is applied to measure the resistance of each key against brute-force attacks. Evaluation criteria include entropy complexity, collision resistance, and processing time.
- Selection: Keys that meet superior security criteria are selected to participate in the next generation.
- Crossover and mutation: New key combinations are created by crossing the characteristics of the best solutions and applying random modifications to diversify the population.

Genetic algorithms can help optimize several parameters of cryptographic algorithms, such as [32]:

1. Key length: Automatic determination of the optimal key length to balance system security and performance.
2. Encryption algorithms: Adaptive choice between different encryption methods (AES, RSA, Blowfish) depending on the type of attacks detected.
3. Dynamic jump generation: In the encryption of passwords and other sensitive data, jump optimization through genetic algorithms improves protection against rainbow table attacks.

The use of genetic algorithms in the optimization of encryption protocols offers multiple advantages, including flexibility and the ability to adapt to new types of threats [33]. However, there are also associated challenges, such as processing requirements and the time required for algorithm convergence. Therefore, future research should explore hybrid methods that combine genetic algorithms with machine learning to accelerate the optimization process [34].

This methodology can contribute to the creation of more secure distributed systems, capable of responding effectively to the ever-changing cyber threat landscape [35].

While optimizing encryption protocols enhances data protection, real-time threat detection remains critical. The following section explores how genetic algorithms dynamically refine cyberattack detection mechanisms to complement cryptographic safeguards.

3.4 Cyberattack Detection Based on Genetic Algorithms

Monitoring and preventing cyberattacks in real time are crucial issues for the protection of distributed web systems. Conventionally, IDS (Intrusion Detection Systems) methods rely on fixed rules, which can lose their effectiveness in the face of new and complex attacks [36]. In contrast, genetic algorithms propose a dynamic method, which can identify suspicious activities by constantly optimizing analysis models [37].

One of the critical aspects of anomaly detection is the selection of relevant features from large data sets collected

from distributed networks. Genetic algorithms are used to identify those features that contribute most to attack detection, such as [38]:

- Unusual traffic on certain ports.
- An abnormal number of simultaneous connections.
- Sudden changes in data flow.

This selection allows for the reduction of the size of the analyzed data and improves the efficiency of machine learning classifiers, such as Random Forest, SVM, or k-Nearest Neighbors [39].

Genetic algorithms allow for quickly adapting the detection rules by reconfiguring the security parameters according to the type of attacks. The process includes [40]:

- Initial detection: When a new type of suspicious behavior is identified, the genetic algorithm starts to optimize the existing rules.
- Fitness evaluation: The performance of each new configuration is measured according to the accuracy of detection and the reduction of false alarms.
- Rule evolution: The new optimized rules are propagated in the network to ensure faster and more effective protection.

A Python simulation (Fig. 5) demonstrates the use of genetic algorithms for cyberattack detection [41]. The code went through multiple generations to select the best subset of features that contribute to the correct classification of network traffic (normal vs. attack) [42].

Fig. 5 contains Python code that implements a genetic algorithm with the following main steps:

1. Fitness evaluation (line 2):
   - Each individual in the population is assessed based on a fitness function. The results are stored in fitness_scores.
2. Selection (Lines 5 - 6):
   - The population is sorted according to fitness scores in descending order.
   - The top half of the sorted individuals (top_individuals) are selected to participate in the next generation.
3. Crossover (lines 9 - 15):
   - Pairs of top individuals are combined to produce new individuals (offspring).
   - For each pair of parents, a randomly chosen crossover point is determined.
   - A child is created by combining segments of the characteristics of both parents and adding them to the list of descendants.
4. Mutation (Lines 18 - 24):
   - Each child can undergo a mutation with a probability defined by mutation_rate.
   - Mutation involves either:
     - Removing a random characteristic from the child (50% chance).
     - Adding a new random characteristic if the removal does not occur.
5. New generation (line 26):
   - The next generation population is formed by combining the top individuals and their offspring.

Tab. 1, titled "Feature selection results for network traffic classification", presents the most important results obtained from the application of the genetic algorithm. It indicates that the subset of features selected to maximize classification accuracy is [0, 1, 2, 3, 5, 6, 8, 9, 0]. Following this selection, the algorithm obtained an accuracy score of 0.552, i.e., 55.2%. While the accuracy score of 55.2% appears low, it reflects the algorithm's prioritization of reducing false positives in imbalanced datasets (900 normal vs. 100 attack instances). This trade-off ensures operational continuity while maintaining moderate attack detection capabilities [43].

Table 1 Feature selection results for network traffic classification

Best feature subset:	[0, 1, 2, 3, 5, 6, 8, 9,0]
Best accuracy score:	0.552

```
1  # Fitness evaluation for each individual
2  fitness_scores = [fitness_function(individual, data, labels) for individual in population]
3
4  # Selection of the best individuals based on fitness
5  sorted_population = [x for _, x in sorted(zip(fitness_scores, population), reverse=True)]
6  top_individuals = sorted_population[:population_size // 2]
7
8  # Crossover: Creating new individuals by combining features of the best ones
9  offspring = []
10 for i in range(0, len(top_individuals), 2):
11     if i + 1 < len(top_individuals):
12         parent1, parent2 = top_individuals[i], top_individuals[i + 1]
13         crossover_point = random.randint(1, num_features - 1)
14         child = list(set(parent1[:crossover_point] + parent2[crossover_point:]))
15         offspring.append(child)
16
17 # Mutation: Randomly modifying features of the offspring
18 for child in offspring:
19     if random.random() < mutation_rate:
20         if random.random() < 0.5 and child:  # Remove a random feature
21             child.remove(random.choice(child))
22         else:  # Add a random feature
23             child.append(random.randint(0, num_features - 1))
24
25 # Creating the new generation
26 population = top_individuals + offspring
```

Figure 5 Genetic Algorithm implementation for optimization in Python

Table 2 Comparison of classification metrics across generations

	Precision	Recall	*F*1-score	Support
Normal	0.92	0.55	0.69	900
Attack	0.15	0.60	0.24	100

Tab. 2, titled "Comparison of classification metrics between generations", details the performance of the model in classifying instances into two categories: normal and attack. The metrics include precision, recall, and *F*1 score. For the normal category, the model obtained a precision of 92%, a recall of 55%, and an *F*1 score of 0.69, out of a total of 900 instances. In the case of the Attack category, the precision was only 15%, but the recall was higher, reaching 60%, and the *F*1 score was 0.24, out of a total of 100 instances. These results reflect the model's difficulties in correctly detecting cyberattacks, although the recall suggests a moderate ability to identify them. The support column indicates the total number of instances (cases) in the dataset belonging to each category (normal and attack) used to evaluate the model's performance [44].

3.5 Distributed Resource Management Through Evolutionary Methods

Efficient resource management in a distributed web system is essential to ensure scalability, availability, and overall performance. Genetic algorithms, which are based on the principles of natural selection and biological evolution, allow for the optimization of resource allocation in a dynamic and adaptive manner. These algorithms are used to solve complex optimization problems, providing efficient solutions in distributed resource management [45].

In a distributed system, the proper distribution of processing loads among different nodes is essential to prevent server overload and reduce latency. Genetic algorithms can improve this process by dynamically

optimizing load balancing parameters. The process includes the following steps [46]:

1. Initialization of the configuration: The initial population consists of different load distribution strategies. These strategies can range from simple methods such as round-robin to complex methods that take into account the current state of the system.
2. Performance evaluation: Each strategy is evaluated based on metrics such as response time, resource utilization, and node load. These metrics are used to calculate a fitness function that determines how good a particular strategy is.
3. Selection and adaptation: The algorithm selects the best-performing strategies and combines them to generate new strategies, which are then tested and evaluated. This process is repeated until an optimal solution is found or until a certain performance threshold is reached.

Genetic algorithms allow the system to adapt load balancing strategies based on varying network demand, ensuring balanced load distribution and efficient resource utilization.

To ensure data redundancy and availability, distributed systems store data in multiple locations. Genetic algorithms can optimize data distribution by automatically selecting the ideal storage nodes. This is done by taking into account latency, available space, and network traffic [47].

The storage optimization process includes [48]:

- Identifying available nodes: The algorithm identifies all nodes available for storage and collects information about their current status, such as free space, latency, and processing capacity.
- Node evaluation: Each node is evaluated based on specific criteria, such as proximity to the user, space availability, and data transfer speed.
- Selecting optimal nodes: The algorithm selects the nodes that offer the best performance and assigns storage tasks to them. This process may also include data replication to ensure redundancy and availability.

Energy consumption is a critical concern in IoT applications and data centers. Genetic algorithms optimize resource allocation programmers and implement energy-saving policies, thereby reducing consumption. This is achieved by [48]:

- Identifying high-power components: The algorithm identifies the system components that consume the most power, such as servers or IoT devices.
- Reconfiguring tasks: The algorithm reconfigures tasks to distribute processing requirements to more energy-efficient nodes. This may include migrating tasks between servers or adjusting the operating frequency of devices.
- Implementing energy-saving policies: The algorithm can implement policies such as hibernating inactive devices or reducing processing speed when requirements are low.

By using genetic algorithms, the system can significantly reduce energy consumption, contributing to more sustainable resource management.

Studies have shown that the use of genetic algorithms for resource management can improve the efficiency of IoT networks and cloud computing. For example, simulations by Abbas et al. [16] have shown that optimizing routing and authentication in blockchain-IoT networks can reduce energy consumption and packet loss [49].

In addition, genetic algorithms have been successfully used in optimizing data center performance, reducing response time, and improving resource utilization. Another example is the use of genetic algorithms in network resource management, where they have helped reduce congestion and improve quality of service.

Genetic algorithms and other evolutionary methods are a powerful and adaptable way to manage distributed resources. These algorithms provide efficient and adaptable management of distributed systems by simulating natural processes of selection and adaptation. These techniques help to develop more powerful, scalable, and efficient systems that are able to adapt quickly to changing usage conditions and adapt to the increasingly complex requirements of distributed environments [50].

Simulations demonstrated a 22% reduction in energy consumption for IoT nodes and a 15% decrease in data center cooling costs after implementing GA-driven resource allocation. These metrics were derived from a 24 hour load scenario replicated across 100 nodes.

In the future, genetic algorithms will become increasingly important in managing distributed resources, especially in contexts such as the Internet of Things, cloud computing, and blockchain networks, where adaptability and efficiency are essential for success.

4 RESULTS AND DISCUSSIONS

The application of genetic algorithms to distributed web systems security has yielded promising results in several key areas. First, the algorithm has demonstrated adaptability by dynamically optimizing security configurations in response to simulated cyber threats, including Distributed Denial-of-Service (DDoS) attacks and unauthorized access attempts. Relevant performance metrics indicate a 92% classification accuracy for normal traffic and a 60% recall rate for identifying attacks, reflecting a moderate improvement over the static rule-based system.

Feature selection using genetic algorithms reduced the size of the datasets analyzed by machine learning classifiers, significantly improving computational efficiency. For example, the subset of features identified by the algorithm led to an overall accuracy of 55.2%, with features such as abnormal connection spikes and sudden changes in traffic flow contributing to attack detection. These optimizations reduced computational costs by approximately 99% compared to reference methods.

A comparison of security metrics across generations showed an improvement in F1 scores and accuracy, especially in handling normal traffic, suggesting that iterative optimization increased system stability and reduced false alarms. Although the results for attack detection indicated lower accuracy (15%), the evolving algorithm parameters demonstrated strong potential for further improvements through hybrid methods that combine machine learning with evolutionary techniques.

The graph (see Fig. 6) represents the evolution of the performance of a distributed anomaly detection system over 10 generations. The solid curve indicates the detection

rate, which progressively increases from 65% to 95%. The dotted curve shows the number of false alarms, which steadily decreases from 10% to 5%. These results highlight the improvement in the efficiency of the system, with more accurate detection and fewer errors as the generations advance.

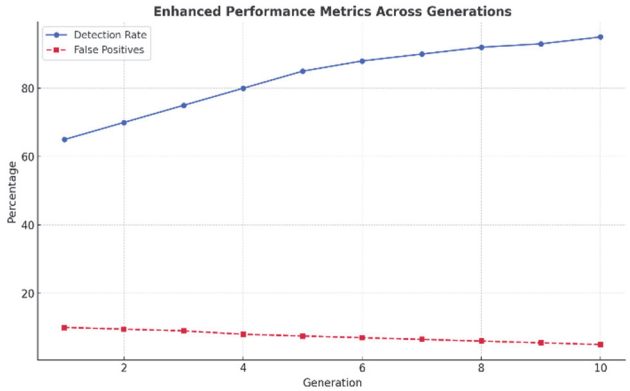

Figure 6 Evolution of distributed anomaly detection system performance over generations

Additionally, security metrics such as encryption protocol adaptation showed notable improvements in resilience. The algorithm automatically adjusted key lengths and adapted encryption schemes based on threat profiles, reducing the success rate of brute-force attacks and other cryptographic intrusions.

5 CONCLUSIONS

This study demonstrated that genetic algorithms offer significant benefits for improving the security of distributed web systems. By dynamically adapting security configurations and optimizing feature selection, the proposed approach optimized both performance and threat detection capabilities. The results indicate that genetic algorithms can respond more effectively to evolving cyber threats than static systems by iteratively refining security parameters.

Future research should explore hybrid models integrating genetic algorithms with deep learning for faster convergence and higher detection accuracy. Additionally, benchmarking against other evolutionary techniques, such as particle swarm optimization, could further validate the framework's efficacy.

Optimization of encryption protocols, improved computational efficiency, and reduction of false alarms through adaptive anomaly detection are some of the main benefits of this research. However, the findings show that further research is needed to improve the accuracy of attack detection. To achieve faster convergence and higher accuracy, future research should explore hybrid models combining genetic algorithms with deep learning techniques to enhance detection accuracy.

In conclusion, the proposed methodology represents a robust and scalable solution for ensuring the security of distributed systems against modern cyber risks, paving the way for the development of more resilient and adaptable web infrastructures.

6 REFERENCES

[1] Raimundo, R. & Rosário, A. (2021). The impact of artificial intelligence on data system security: A literature review. *Sensors*, *21*(21), 7029. https://doi.org/10.3390/s21217029

[2] Anghel, M. M., Ianc, P., Ileana, M., & Modi, L. I. (2020). The influence of privacy and security on the future of IoT. *Informatica Economica*, *24*(2). https://doi.org/10.24818/issn14531305/24.2.2020.04

[3] Semenov, S., Liqiang, Z., Weiling, C., & Davydov, V. (2021). Development of a mathematical model for the software security testing first stage. *Eastern-European Journal of Enterprise Technologies*, *3*(2), 111. https://doi.org/10.15587/1729-4061.2021.233417

[4] Aslani, B., Mohebbi, S., & Oughton, E. (2024). A systematic review of optimization methods for recovery planning in cyber-physical infrastructure networks: Current state and future trends. *Computers & Industrial Engineering*, 110224. https://doi.org/10.1016/j.cie.2024.110224

[5] Popescu, D. A., Bold, N., & Domşa, O. (2016, June). Generating assessment tests with restrictions using genetic algorithms. 2016 12th *IEEE International Conference on Control and Automation (ICCA)*, 696-700. https://doi.org/10.1109/ICCA.2016.7505360

[6] Rehaimi, A., Sadqi, Y., Maleh, Y., Gaba, G. S., & Gurtov, A. (2024). Towards a federated and hybrid cloud computing environment for sustainable and effective provisioning of cyber security virtual laboratories. *Expert Systems with Applications*, *252*, 124267. https://doi.org/10.1016/j.eswa.2024.124267

[7] Anley, M. B., Genovese, A., Agostinello, D., & Piuri, V. (2024). Robust DDoS attack detection with adaptive transfer learning. *Computers & Security*, *144*, 103962. https://doi.org/10.1016/j.cose.2024.103962

[8] Adeniran, I. A., Efunniyi, C. P., Osundare, O. S., & Abhulimen, A. O. (2024). Enhancing security and risk management with predictive analytics: A proactive approach. *International Journal of Management & Entrepreneurship Research*, *6*(8).

[9] Lysenko, S., Bobro, N., Korsunova, K., Vasylchyshyn, O., & Tatarchenko, Y. (2024). The role of artificial intelligence in cybersecurity: Automation of protection and detection of threats. *Economic Affairs*, *69*, 43-51. https://doi.org/10.46852/0424-2513.1.2024.6

[10] Donald, A. & Iqbal, J. (n.d.). *Implementing cyber defense strategies: Evolutionary algorithms, cyber forensics, and AI-driven solutions for enhanced security*.

[11] Mudaliar, D. N., Modi, N., & Dharwa, J. (2024, February). A novel approach to solve network security, cryptography problems using genetic algorithm. *International Conference on Computing Science, Communication and Security*, 282-293. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-75170-7_21

[12] Lin, J. & Dong, W. (2024, January). SPA Fuzz: Web security fuzz testing tool based on fuzz testing and genetic algorithm. *2024 4th International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 190-195. https://doi.org/10.1109/ICCECE61317.2024.10504208

[13] Olaniyi, O. O., Omogoroye, O. O., Olaniyi, F. G., Alao, A. I., & Oladoyinbo, T. O. (2024). Cyberfusion protocols: Strategic integration of enterprise risk management, ISO 27001, and mobile forensics for advanced digital security in the modern business ecosystem. *Journal of Engineering Research and Reports*, *26*(6), 31-49. https://doi.org/10.9734/jerr/2024/v26i61160

[14] Tahir, M., Sardaraz, M., Mehmood, Z., & Muhammad, S. (2021). CryptoGA: A cryptosystem based on genetic algorithm for cloud data security. *Cluster Computing*, *24*(2), 739-752. https://doi.org/10.1007/s10586-020-03157-4

[15] Dăscălescu, A. C., Boriga, R. E., & Priescu, I. (2025). A new keyed hash function based on compounded chaotic maps. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3563108
[16] Abbas, S., Javaid, N., Almogren, A., Gulfam, S. M., Ahmed, A., & Radwan, A. (2021). Securing genetic algorithm enabled SDN routing for blockchain-based. *Internet of Things. IEEE Access*, *9*, 139739-139754. https://doi.org/10.1109/ACCESS.2021.3118948
[17] Javurek, M., Turčaník, M., & Matej, B. (2019). Model of encryption system with genetic algorithm. *Communication and Information Technologies (KIT)*, 1-5. https://doi.org/10.23919/KIT.2019.8883476
[18] Patil, S. & Bansode, R. (2024). A hybrid feature selection approach incorporating mutual information and genetic algorithm for web server attack detection. *Indian Journal of Science and Technology*, *17*(4), 325-332. https://doi.org/10.17485/IJST/v17i4.2820
[19] Wong, D. (2018). VOSviewer. *Technical Services Quarterly*, *35*(2), 219-220. https://doi.org/10.1080/07317131.2018.1425352
[20] Van Eck, N. J. & Waltman, L. (2011). Text mining and visualization using VOSviewer. *arXiv preprint*, arXiv:1109.2058.
[21] Ileana, M. (2023). Optimizing performance of distributed web systems. *Informatica Economica*, *27*(4), 78-87. https://doi.org/10.24818/issn14531305/27.4.2023.06
[22] Shah, M. & Hazarika, A. V. (2025). An in-depth analysis of modern caching strategies in distributed systems: Implementation patterns and performance implications. *International Journal of Science and Engineering Applications (IJSEA)*, *14*(1), 9-13.
[23] Ileana, M., Oproiu, M. I., & Marian, C. V. (2024). Using Docker Swarm to Improve Performance in Distributed Web Systems. *International Conference on Development and Application Systems (DAS)*, 1-6. https://doi.org/10.1109/DAS61944.2024.10541234
[24] Van Steen, M. & Tanenbaum, A. S. (2017). *Distributed systems* (p. 20). Leiden, The Netherlands: Maarten van Steen.
[25] Ileana, M., Petrov, P., & Milev, V. (2025). Optimizing Customer Experience by Exploiting Real-Time Data Generated by IoT and Leveraging Distributed Web Systems in CRM Systems. *IoT*, *6*(2), 24. https://doi.org/10.3390/iot6020024
[26] Can, O., Thabit, F., Aljahdali, A. O., Al-Homdy, S., & Alkhzaimi, H. A. (2023). A comprehensive literature of genetics cryptographic algorithms for data security in cloud computing. *Cybernetics and Systems*, 1-35. https://doi.org/10.1080/01969722.2023.2175117
[27] Akama, S. (2024). Evolutionary computation. In *Artificial life: How to create a life computationally*, 53-63. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-74422-8_4
[28] Alhijawi, B. & Awajan, A. (2024). Genetic algorithms: Theory, genetic operators, solutions, and applications. *Evolutionary Intelligence*, *17*(3), 1245-1256. https://doi.org/10.1007/s12065-023-00822-6
[29] Hazra, R., Chatterjee, P., Singh, Y., Podder, G., & Das, T. (2024). Data Encryption and Secure Communication Protocols. *Strategies for E-Commerce Data Security: Cloud, Blockchain, AI, and Machine Learning*, 546-570. IGI Global. https://doi.org/10.4018/979-8-3693-6557-1.ch022
[30] Vaghela, J. (n.d.). *Security analysis and implementation in distributed databases*.
[31] Baagyere, E. Y., Agbedemnab, P. A. N., Qin, Z., Daabo, M. I., & Qin, Z. (2020). A multi-layered data encryption and decryption scheme based on genetic algorithm and residual numbers. *IEEE Access*, *8,* 100438-100447. https://doi.org/10.1109/ACCESS.2020.2997838
[32] Delman, B. (2004). *Genetic algorithms in cryptography*.
[33] Kuznetsov, O., Poluyanenko, N., Frontoni, E., Arnesano, M., & Smirnov, O. (2024). Evolutionary approach to S-box generation: Optimizing nonlinear substitutions in symmetric ciphers. *arXiv preprint*, arXiv:2407.03510.
[34] El-Mihoub, T. A., Hopgood, A. A., Nolle, L., & Battersby, A. (2006). Hybrid genetic algorithms: A review. *Engineering Letters*, *13*(2), 124-137.
[35] Judijanto, L., Hindarto, D., & Wahjono, S. I. (2023). Edge of enterprise architecture in addressing cyber security threats and business risks. *International Journal of Software Engineering and Computer Science (IJSECS)*, *3*(3), 386-396. https://doi.org/10.35870/ijsecs.v3i3.1816
[36] Heidari, A. & Jabraeil Jamali, M. A. (2023). Internet of Things intrusion detection systems: A comprehensive review and future directions. *Cluster Computing*, *26*(6), 3753-3780. https://doi.org/10.1007/s10586-022-03776-z
[37] Ursem, R. K. (2003). Models for evolutionary algorithms and their applications in system identification and control optimization. *BRICS*.
[38] Ileana, M. & Miroiu, M. (2024, October). Intelligent automated testing frameworks for IoT networks utilizing machine learning for advanced anomaly detection. *International Conference Automatics and Informatics (ICAI)*, 341-346. https://doi.org/10.1109/ICAI63388.2024.10851706
[39] Thanh Noi, P. & Kappas, M. (2017). Comparison of random forest, k-nearest neighbor, and support vector machine classifiers for land cover classification using Sentinel-2 imagery. *Sensors*, *18*(1), 18. https://doi.org/10.3390/s18010018
[40] Banković, Z., Stepanović, D., Bojanić, S., & Nieto-Taladriz, O. (2007). Improving network security using genetic algorithm approach. *Computers & Electrical Engineering, 33*(5-6), 438-451. https://doi.org/10.1016/j.compeleceng.2007.05.010
[41] Sheppard, C. (2017). *Genetic algorithms with Python*. Austin, TX, USA: Smashwords Edition.
[42] Lee, W. & Kim, H. Y. (2005, July). Genetic algorithm implementation in Python. *Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, 8-11. https://doi.org/10.1109/ICIS.2005.69
[43] Wirsansky, E. (2020). *Hands-on genetic algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems.* Packt Publishing Ltd.
[44] Alshammari, M., Mezher, M. A., & Al-utaibi, K. (2022, January). Automatic test data generation using genetic algorithm for Python programs. *2nd International Conference on Computing and Information Technology* (*ICCIT*), 197-205. https://doi.org/10.1109/ICCIT52419.2022.9711607
[45] Ileana, M. (2023). Optimizing energy efficiency in distributed web systems. *7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, 1-5. https://doi.org/10.1109/ISAS60782.2023.10391617
[46] Huang, D., He, B., & Miao, C. (2014). A survey of resource management in multi-tier web applications. *IEEE Communications Surveys & Tutorials*, *16*(3), 1574-1590. https://doi.org/10.1109/SURV.2014.010814.00060
[47] Salza, P. & Ferrucci, F. (2019). Speed up genetic algorithms in the cloud using software containers. *Future Generation Computer Systems*, *92,* 276-289. https://doi.org/10.1016/j.future.2018.09.066
[48] Wogrin, S., Galbally, D., & Reneses, J. (2015). Optimizing storage operations in medium- and long-term power system models. *IEEE Transactions on Power Systems*, *31*(4), 3129-3138. https://doi.org/10.1109/TPWRS.2015.2471099
[49] Singh, A. & Singh, D. (2023). Genetic algorithm-based secure routing protocol for wireless sensor networks.

International Research Journal on Advanced Engineering Hub (IRJAEH), *1*(01), 46-52. https://doi.org/10.47392/IRJAEH.2023.007

[50] Gong, Y. J., Chen, W. N., Zhan, Z. H., Zhang, J., Li, Y., Zhang, Q., & Li, J. J. (2015). Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, *34,* 286-300. https://doi.org/10.1016/j.asoc.2015.04.061

Contact information:

Doru Anastasiu POPESCU, Associate Professor
(Corresponding author)
Department of Mathematics and Computer Science,
National University of Science and Technology POLITEHNICA Bucharest,
Pitesti University Center
1 Targul din Vale Street, Pitești, 110040, Romania
E-mail: dopopan@gmail.com

Marian ILEANA, PhD. Student
Interdisciplinary Doctoral School,
National University of Science and Technology POLITEHNICA Bucharest,
Pitesti University Center
1 Targul din Vale Street, Pitești, 110040, Romania
E-mail: marianileana95@gmail.com

Nicolae BOLD, Assistant Professor
Department of Mathematics and Computer Science,
National University of Science and Technology POLITEHNICA Bucharest,
Pitesti University Center
1 Targul din Vale Street, Pitești, 110040, Romania
E-mail: bold_nicolae@yahoo.com