BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 74(2), 2026, Article number: e157325
DOI: 10.24425/bpasts.2026.157325

An evaluation of an artificial immune system based approach to anomaly detection in computer programs

Patryk WIDULIŃSKI*, Krzysztof WAWRYN

Koszalin University of Technology, Faculty of Electronics and Computer Science, Śniadeckich 2, 75-453 Koszalin, Poland

Abstract. The growing sophistication of cyber threats and the limitations of traditional intrusion detection systems (IDS) have led researchers to explore biologically inspired models. One promising approach involves the application of artificial immune systems (AIS), which mimic the self/nonself discrimination mechanism of biological immune systems. In this paper, we propose an IDS based on the negative selection algorithm (NSA), enhanced by a novel modification involving intercellular receptors (ICRs). This dual-receptor architecture improves detection accuracy by targeting both standard and intercellular anomalies in program code. We present a mathematical model of the system, describe its implementation, and evaluate its performance across three key metrics: detection rate, memory efficiency, and processing speed. Experimental results demonstrate that the modified NSA with ICRs matches or outperforms existing methods, achieving an average detection accuracy improvement of 8.1%.

Keywords: intrusion detection system; malware; anomaly detection; negative selection

1. INTRODUCTION

Over the past few decades, the prevalence of computers and computer networks in the public space has initiated a challenge: ensuring their security against a multitude of cyber threats. Intrusion detection systems (IDS) have emerged as a line of defense, aimed at identifying and eliminating potential infections that compromise the integrity of these systems. However, with the evolving nature of malicious attacks and the increasing complexity of computer programs, there is a growing demand for innovative approaches that can effectively counter these emerging threats.

In recent years, researchers have turned to artificial immune systems (AIS) for inspiration in designing IDS solutions that mimic the powerful defense mechanisms of biological immune systems. An important concept within the domain of AIS is the negative selection algorithm (NSA), which draws upon the principles of self/nonself discrimination to identify anomalies. By constructing a set of binary strings called receptors (or detectors), the IDS can then compare them against monitored software fragments, aiming to detect deviations that may indicate the presence of an intrusion. Information on the fundamentals of IDS based on AIS can be found in [1–6]. These foundational studies have paved the way for more advanced systems discussed in [7–14].

1.1. NSA approach review

Approaches based on NSA have become very popular when it comes to the task of detecting intrusions. Ji, Dasgupta [15] improved NSA with variable-length detectors (V-detectors). These detectors, thanks to their varying length, better "plug" the holes created during generation. Their research showed that the algorithm performance improved without a large increase in complexity. In [16], a MILA (multilevel immune learning algorithm) system was proposed, which combines the use of NSA with receptor expansion via a clonal method and dynamic receptor generation in a unified solution. Li *et al*. [17] proposed FB-NSA, a variant of NSA which has two types of receptors: constant-sized receptors (CFB-NSA) and variable-sized receptors (VFB-NSA). Bejoy *et al*. [18] developed a generic framework for an artificial immune system for detecting anomalies, addressing the problems of overlapping and holes. In [19], a multiclass anomaly detection algorithm hybridizing NSA and the clonal selection algorithm (CSA) was proposed. In [20], the authors proposed a real-valued NSA based on hierarchy division (HD-NSA). In [21], the NSA was used to detect unknown malware in the Internet of Things (IoT) environment. In [22], a comprehensive review of the recent use of the NSA was provided. A local intrusion detection system based on NSA was proposed in [23–27]. The NSA can also be used not only for malware detection but also, for example, for steganography, which has been described in [28, 29]. A combination of the blockchain and artificial immune systems is also viable [30]. Application of the principles of artificial immune systems is also possible in areas other than intrusion detection. Xuan *et al*. [31] proposed the artificial-immune-based AIDE algorithm to solve the distributed heterogeneous flexible flowshop problem (DHFFP) in the steel production process.

Broadly, these contributions address three recurring challenges: improving coverage of the nonself space (V-detectors, FB-NSA, HD-NSA), managing resource usage and detector generation cost (MILA, generic AIS frameworks), and adapting NSA to specific domains such as networks, IoT, or industrial optimization. However, almost all of these approaches operate

*e-mail: patryk.widulinski@tu.koszalin.pl

Manuscript submitted 2024-10-07, revised 2025-12-03, initially accepted for publication 2025-12-05, published in March 2026.

on contiguous feature regions. They do not explicitly model subtle modifications that arise between adjacent memory cells in compiled program code – a scenario where byte-level anomalies may span across cell boundaries and therefore evade receptors that only observe aligned fragments.

To address this gap, this paper proposes an enhancement to the NSA that introduces intercellular receptors (ICRs), enabling the system to monitor regions between memory cell boundaries. The ICR mechanism is designed to increase detection accuracy for small, non-aligned binary anomalies while preserving performance efficiency, particularly in low-memory or time-sensitive environments.

1.2. The aim of the work

This paper proposes an IDS that utilizes the potential of the NSA to detect infections (modifications) in computer programs. The system enhances detection capabilities by incorporating a custom modification of the algorithm. Through the use of an additional set of intercellular receptors (ICR), the proposed IDS identifies modifications in computer programs with increased accuracy. The paper is organized as follows. Section 2 presents the mathematical principles of the algorithm. Section 3 describes the implementation of the intrusion detection system. Section 4 outlines the proposed algorithm modification. Section 5 provides experimental results and comparisons with existing solutions. Finally, Section 6 concludes the work.

2. MATHEMATICAL MODEL

2.1. The negative selection algorithm

The NSA is an algorithm inspired by the human immune system's ability to distinguish between self and nonself cells. NSA is a type of machine learning algorithm used in anomaly detection applications, such as intrusion detection systems. NSA works by generating a set of receptors that can recognize nonself (anomalous) patterns. The receptors are trained on a set of self (normal) patterns, which represent the typical behavior of the system being monitored. The self patterns are usually represented as strings of bits, and the receptors as subsets of the possible strings. The receptor set typically covers most of the nonself (anomalous) patterns. In the intrusion detection context, NSA generates a set of receptors, **R**, based on the normal behavior of the system. It then compares the current version of the monitored data with the receptor set to identify anomalous behavior. When a match is found between a receptor and monitored data, the algorithm flags it as anomalous. The proposed IDS algorithms are based on NSA principles, which are formally described in the following subsections.

2.2. Binary string representation of self and nonself patterns

The goal of the method is to find a receptor set **R** that matches as many of the nonself strings in **N** as possible, without matching any of the self strings in **S**. The sets of binary strings and their relations are depicted in Fig. 1. Given an l-dimensional Hamming space $\mathbf{Z}_l = \mathbf{U}$, where l is a parameter that defines the length of a cell (i.e., a self string), and a given set of self strings $\mathbf{S} \subseteq \mathbf{U}$, the set of string-receptors $\mathbf{R} \subseteq \mathbf{U}$ satisfying the condition of "self-nonself recognizability" should be constructed. For the purpose of clarity, the strings which do not satisfy this condition are denoted $\mathbf{R}' \subseteq \mathbf{U}$, and are called rejected receptor candidates. For a given parameter $m \leq l$, called the threshold of activation, and for any string $a \in \mathbf{U}$, let us define $a^{(k,m)}$ as an m-element subsequence of the string a starting from position k. The first part of the condition ("self") can be formulated in the following way:

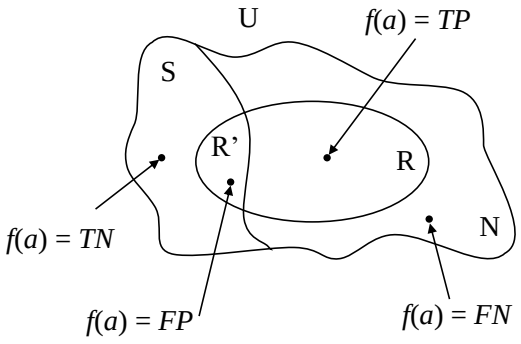

Fig. 1. The sets of binary strings and system prediction results f(a) for a given binary string a (dots).

$$\forall s \in \mathbf{S}, r \in \mathbf{R}, k \in \{1,...,l-m+1\} : r^{(k,m)} \neq s^{(k,m)}. \quad (1)$$

This means that no receptor will be activated when paired with one of the self strings. The corresponding "nonself" condition for a given set of nonself strings $\mathbf{N} \subseteq \mathbf{U}$ can be formulated as follows:

$$\forall n \in \mathbf{N}, \exists r \in \mathbf{R}, k \in \{1,...,l-m+1\} : r^{(k,m)} = n^{(k,m)}. \quad (2)$$

This means that each nonself string n activates at least one receptor. In view of the formation of a proper relationship between the effectiveness of protection and its cost, it is postulated that the condition of self/nonself recognition is satisfied for the smallest possible set **R** and the largest possible set **N**.

2.3. The outcome of the system

Let $f(a)$ be the outcome function of the system, where $a \in \mathbf{U}$ is a binary string tested for anomalies. Then:

$$f(a) = \begin{cases} \text{TP} & \text{if } a \in \mathbf{N}, \exists r \in \mathbf{R} : match(r,a) \\ \text{FP} & \text{if } a \in \mathbf{S}, \exists r \in \mathbf{R} : match(r,a) \\ \text{TN} & \text{if } a \in \mathbf{S}, \nexists r \in \mathbf{R} : match(r,a) \\ \text{FN} & \text{if } a \in \mathbf{N}, \nexists r \in \mathbf{R} : match(r,a) \end{cases} \quad (3)$$

The match between a receptor r and a tested string a is denoted $match(r,a)$. The system uses receptor set **R** to predict whether the tested string is anomalous. The prediction can be either a negative ("N", no anomaly) or a positive ("P", anomaly predicted). According to $f(a)$, whenever the system predicts correctly it reports a true result ("T"), and a false result ("F")

otherwise. Thus, the system reports a true positive ("TP") when it predicts an anomaly and the prediction is accurate. Similarly, the system reports a false positive ("FP") when it predicts an anomaly but the prediction is wrong, i.e., the tested string belongs to **S**. In case a tested string is predicted to be a normal (self) string, and is indeed a normal string (belongs to **S**), then the system outcome is a true negative ("TN"). In case of an incomplete training set, false negatives ("FN") may occur. This is when the system incorrectly predicts that the tested string is normal (self), when it actually belongs to **N**. The incompleteness of the training set implies $\mathbf{R}' \neq \emptyset$.

3. THE INTRUSION DETECTION SYSTEM

The proposed IDS setup has been depicted in Fig. 2. It comprises several components: a monitored directory (MD), a receptor set ($\mathbf{R} + \mathbf{R_I}$), a receptor generation unit (RGU), an anomaly detection unit (ADU), a control unit (CU), and interface blocks (I/F). The MD is a directory containing programs to be scanned cyclically by the IDS for possible infections. For instance, in Windows, the C:\Windows directory houses important executable files required for the operating system to function. This directory could become the monitored directory in a practical setup.

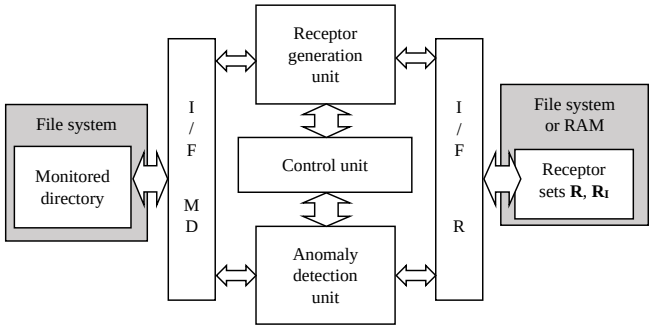

Fig. 2. A diagram of the proposed intrusion detection system

The main unit of the system is the CU. It is responsible for invoking the RGU first (which generates the receptors) for each monitored file in the MD, and subsequently invoking the ADU to scan for anomalies using the generated receptors. The sets **R** and $\mathbf{R_I}$ represent receptors *specific to each program* in the MD. These sets can be placed in either volatile or non-volatile memory, according to the administrator's preference. The I/F blocks allow for adaptive operating-system-based communication between the IDS, the MD, and the read/write operations for **R** and $\mathbf{R_I}$.

The RGU utilizes two methods for generating the receptors: a template method (without our modification) [32], and a modified method that employs intercellular receptors to detect anomalies between program cells - henceforth referred to as the ICR method.

The term *activation* refers to an event in which a receptor candidate matches a self program fragment (during generation) or when a receptor matches an anomaly during anomaly detection. In the former case, the receptor candidate is disregarded from further use. In the latter, the receptor is exhibiting desired behavior.

In Fig. 3, an example of a match between a self fragment and a receptor candidate is shown during the process of receptor generation in RGU. The receptor candidate in this case was C68D040Ah, and it was compared to the 4-byte memory cell under the address 19FC58h. The self fragment under that address was 3B6D0418h. It can be seen that by shifting the activation window by $k = 11$ bits, a match of $m = 16$ consecutive bits was found. This means that the receptor candidate was activated for this pair of compared values and was therefore disqualified from becoming a valid receptor, as it matched a self fragment.

An activation in the ADU indicates that an anomaly has occurred, because activation can only take place between self fragments and receptors. This event is reported in the logs of the IDS.

4. THE MODIFICATION OF THE NSA ALGORITHM

4.1. The motivation

Let us consider a scenario where modifications (anomalies) occur in a computer program. Machine code is the language used to write computer program instructions that are contained in executable files. A typical executable program in the operating system has its instructions presented as shown in Fig. 4. The addresses of the individual bytes of these instructions in the file are split into a base part and an offset to make them easier to read/locate. To make up a specific address, these parts must be added together. For instance, at base address 19FC50h + offset 0h (that is, 19FC50h), a byte with value 08h is present, and at base address 19FC60h + offset Dh (that is, 19FC6Dh), value C7h resides.

Let us consider a scenario where a small-size anomaly occurs between self program fragments of length $l = 32$ (4 bytes), as shown in Fig. 5. To enhance the readability of the figure, the self program fragments have been marked using thick borders. The anomaly introduced by an intruder is marked in red and bold font, and the resulting potential 16-bit intercellular fragment is colored orange. Since the program's self fragments always have the same length as the receptors (in this case, $l = 32$), it is possible that this specific anomaly (the value in red with effective value 7A8Ch) may not be detected by any 32-bit receptor – either at 19FC60h, where the modified digit Ah is missed, or at 19FC64h, where the modified digit 8h is missed.

4.2. The modification

The problem of detecting small-size anomalies between self program fragments, described above, motivated the proposed modification to the NSA.

To increase the coverage of the nonself string space, we propose a modification of the NSA algorithm that deploys additional smaller receptors called intercellular receptors (ICRs). Given the sets **S** and **N**, the intercellular receptor set $\mathbf{R_I} \subseteq \mathbf{U}$ can be constructed. $\mathbf{R_I}$ is an additional (auxiliary) receptor set which is generated using a second set of parameters: l_I, the length of the receptors in $\mathbf{R_I}$, and m_I, their activation threshold. As with **R**, the construction of $\mathbf{R_I}$ similarly relies on the condition of "self-nonself recognizability". For a given parameter $m_I \leq l_I$,

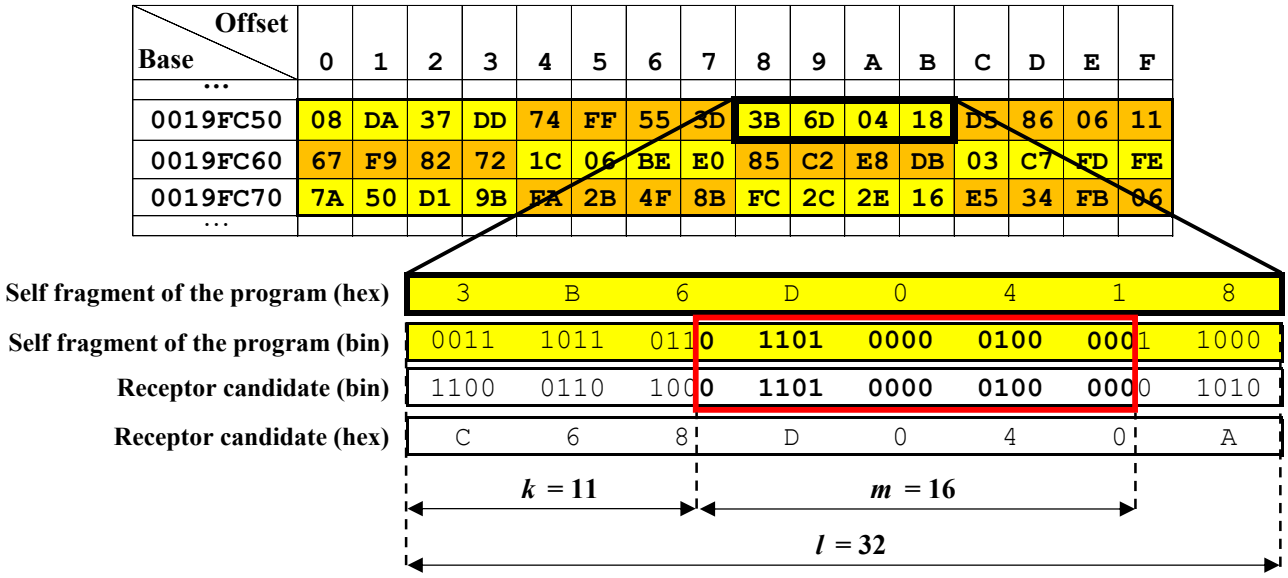

Fig. 3. An example of a match between a self fragment and a receptor candidate during receptor generation. The red outline shows the 16-bit match found by shifting the activation window across the self fragment (3B6D0418) and the receptor candidate (C68D040A) by 11 bits. Matching values are shown in both hexadecimal and binary for clarity

Offset / Base	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
…																
0019FC50	08	DA	37	DD	74	FF	55	3D	3B	6D	04	18	D5	86	06	11
0019FC60	67	F9	82	72	1C	06	BE	E0	85	C2	E8	DB	03	C7	FD	FE
0019FC70	7A	50	D1	9B	FA	2B	4F	8B	FC	2C	2E	16	E5	34	FB	06
…																

Fig. 4. A segment of memory showing program instructions (machine code) as hexadecimal values

Offset / Base	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
…																
0019FC50	08	DA	37	DD	74	FF	55	3D	3B	6D	04	18	D5	86	06	11
0019FC60	67	F9	82	7A	8C	06	BE	E0	85	C2	E8	DB	03	C7	FD	FE
0019FC70	7A	50	D1	9B	FA	2B	4F	8B	FC	2C	2E	16	E5	34	FB	06
…																

Fig. 5. A memory segment containing a subtle anomaly introduced between two 32-bit self fragments, at address 19FC63. Self fragments are outlined in thick borders for visibility. The injected anomaly is shown in red, and the resulting 16-bit intercellular fragment - used by the ICR mechanism - is marked in orange

and for any string $b \in \mathbf{U}$, let us define $b^{(k,m_I)}$ as an m_I-element subsequence of the string b starting from position k. The "self" part of the condition can be formulated as follows:

$$\forall s \in \mathbf{S}, r \in \mathbf{R_I}, k \in \{1,\ldots,l_I - m_I + 1\} : r^{(k,m_I)} \neq s^{(k,m_I)}. \quad (4)$$

This ensures that no auxiliary receptor will be activated when compared with one of the self strings. The "nonself" condition for a given set of nonself strings $\mathbf{N} \subseteq \mathbf{U}$ can be expressed as:

$$\forall n \in \mathbf{N}, \exists r \in \mathbf{R_I}, k \in \{1,\ldots,l_I - m_I + 1\} : r^{(k,m_I)} = n^{(k,m_I)}. \quad (5)$$

$\mathbf{R_I}$ is generated by RGU during the receptor creation stage based on the program fragments located *between* the 32-bit self program cells. The parameters used to discuss the modification are $l = 32$ and $m = 16$. Figure 5 shows, using thicker borders, the locations of 16-bit program fragments used to generate the $\mathbf{R_I}$ set. The $\mathbf{R_I}$ set has its own parameters, l_I and m_I, analogous to l and m. Each 16-bit intercellular fragment consists of the least significant byte of the original 32-bit fragment, which becomes the most significant byte, and the most significant byte of the next 32-bit fragment, which becomes the least significant. For example, a 16-bit intercellular fragment of E085h was created at address 19FC67h, and a fragment of FE7Ah was created at address 19FC6Fh.

The $\mathbf{R_I}$ set is used by ADU at the anomaly detection stage to check for anomalies *between* l-bit cells. However, it is important to note that the $\mathbf{R_I}$ set is only suitable for verifying l_I-bit intercellular fragments, similar to the receptor set $\mathbf{R}$, which is only suitable for scanning l-bit consecutive program fragments. The inclusion of the $\mathbf{R_I}$ set in the detection process increases the probability of detecting an infection. The positive effect of the inclusion of $\mathbf{R_I}$ on detection rates will be demonstrated in the next section, as well as drawbacks.

5. EXPERIMENTAL STUDIES

5.1. Methodology

The IDS, along with the ICR method, was implemented and thoroughly tested for a wide range of input parameters. For testing, the IDS was implemented in C++ and compiled using the MSVC compiler included in the Visual Studio 2019 integrated development environment. All tests were conducted on a workstation with the configuration listed in Table 1.

Table 1
Test workstation configuration

Operating system	Windows 10
OS architecture	64-bit
CPU Intel Core	i9-10900K
CPU clock	3.7 GHz
CPU core count	10
Threads per core	2
RAM size	16 GB
Non-volatile storage type	NVMe SSD

A folder named *ids* was created on the C partition containing the operating system. This location was set in the intrusion detection system as the location to monitor (C:\ids). Since IDS monitors all files in the given folder, it was necessary to obtain a file to monitor. To achieve this, a sample program was written in C++. This program was also compiled using Visual Studio 2019 environment and MSVC compiler. The properties of the monitored program can be found in Table 2.

Table 2
Monitored file properties

File name	test.exe
File size	9216 B
Program format	Win32 Portable Executable (PE)
Target platform	32-bit

IDS testing is based on changing input parameters and checking the effects on the system. Changing the input parameters affects the system's performance characteristics, which makes several evaluation criteria worth noting. The main three evaluation criteria are:

1. The criterion of maximization of detection rate – assuming that an important aspect of IDS operation is to achieve the maximum detection rate of anomalies, even at the expense of memory occupancy or speed.
2. The criterion of memory occupancy minimization – assuming that an important aspect of IDS operation is to achieve a low occupancy of non-volatile or working memory.
3. The criterion of maximizing speed of operation – considering the shortest possible execution time of the algorithms as an aspect more important than the other two.

Figure 6 illustrates the relationship between the evaluation criteria. An IDS user may assume that they are interested in the highest possible anomaly detection. By nature, this is likely to result in higher memory usage and/or slower system performance. Similarly, prioritizing another criterion may negatively impact other results.

The evaluation process for a single test is shown below as a list of steps:

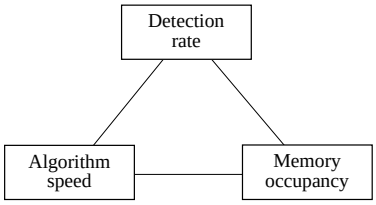

Fig. 6. The relationship between the main criteria for evaluation of the performance of the IDS

1. Save a backup of the test.exe file,
2. Select new l, m (W) and new l_I, m_I (P),
3. Generate new receptor sets for test.exe: $\mathbf{R}$ for $W(l,m)$ and $\mathbf{R_I}$ for $P(l_I, m_I)$,
4. Inject a single 4-byte anomaly into test.exe at a random location,
5. Run the anomaly detection unit,
6. Restore the test.exe file from the backup,
7. Repeat steps 4–6 ninety-nine more times for a total of 100 detection attempts,
8. Repeat the entire process from step 2 until all W and P variants have been evaluated.

The test range $\mathbf{L}$ of receptor length l was:

$$\mathbf{L} = \{16, 17, 18, ..., 32\}, l \in \mathbf{L}.$$

The test range $\mathbf{M}$ of activation threshold m was:

$$\mathbf{M} = \{8, 9, ..., 12\}, m \in \mathbf{M}.$$

The length of the intercellular receptors l_I was set to 16, as this was the lowest possible number of bits covering the full extreme bytes of adjacent memory cells ($2 \cdot 8$). The test range for the activation threshold of m_I intercellular receptors was:

$$\mathbf{M_I} = \{8, 9, 10, 11\}, m_I \in \mathbf{M_I}.$$

All combinations of l, m, l_I, m_I must be tested, so the number of tests performed was:

$$|\mathbf{L}| \cdot |\mathbf{M}| \cdot |\mathbf{M_I}| = 17 \cdot 5 \cdot 4 = 340.$$

Since including all 340 test results is not practical for a journal paper, 10 tests were selected for each criterion to represent the broader results. For each criterion, the tests were sorted by specific columns. The columns in the results are as follows: # – test number, W – the values of l and m, P – the values of l_I and m_I, $TP\%$ – true positive ratio for the non-modified method, $TPI\%$ – true positive ratio for the modified ICR method, $G\%$ – gain of true positive ratio over the non-modified NSA method, Rm – number of non-modified receptors in the $\mathbf{R}$ set in memory, Rm_{ICR} – number of receptors in the ICR receptor set $\mathbf{R_I}$, Rm_T – total number of receptors in memory, $Rm_T\%$ – the ratio of total receptor memory occupancy to program size, TGt – template generation time [ms], RGt – receptor generation time [ms], $RIGt$ – intercellular receptor set generation time [ms], Gt – total generation time [ms].

5.2. The criterion of detection rate maximization

The criterion for maximizing detection rate involves achieving the highest possible $TPI\%$. The results are shown in Table 3. The test results are sorted first by the highest $TPI\%$ value, followed by the lowest memory occupancy ($Rm_T\%$).

Table 3
Test results when considering the maximization of detection rate

#	W	P	$TP\%$	$TPI\%$	$G\%$	$Rm_T\%$
1	20, 10	16, 10	86	95	9	13.3
2	21, 11	16, 10	91	95	4	24.6
3	32, 10	16, 9	93	95	2	27.4
4	17, 12	16, 8	95	95	0	54.9
5	17, 12	16, 10	95	95	0	56.1
6	28, 10	16, 10	89	94	5	21.8
7	32, 10	16, 8	94	94	0	24.5
8	31, 10	16, 11	78	94	16	32.9
9	30, 10	16, 11	91	94	3	37.3
10	18, 10	16, 11	51	93	42	21.4

Figure 7 depicts a plot of true positive ratio ($TPI\%$) and ICR gain ($G\%$) as a function of test number and W for the detection criterion. The results suggest that the ICR method achieves the best results at even values of l and for values of m less than or equal 10. For $m > 10$, in tests 1–3, 42–45, 74, 101–105, an increase in detection rates can be observed, especially for values of l divisible by 4. Higher gains were obtained for $m \leq 10$: The ICR method pays off for even values of the parameter l with the lowest possible parameter m and the highest possible parameter m_I (not depicted to reduce clutter).

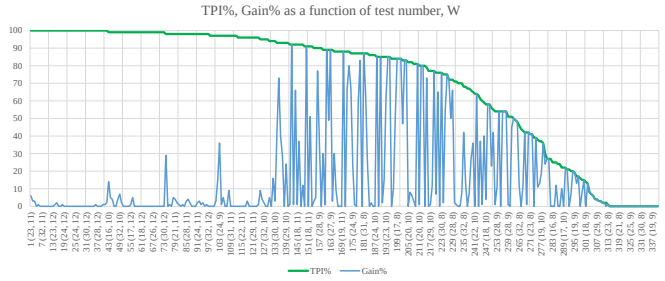

Fig. 7. True positive intercellular ratio [%] and Gain ratio [%] as a function of test number and W – the criterion of maximization of detection rate – all 340 records

5.3. The criterion of minimization of memory occupancy

The criterion for minimization of memory occupancy is to prioritize the lowest possible $Rm_T\%$ parameter while maintaining acceptable detection parameters as defined by the user. The results are shown in Table 4 and are sorted first by the lowest $Rm_T\%$ value.

Figure 8 shows the test results for the memory occupancy minimization criterion. The test results indicate that the ICR method improved detectability most effectively at low memory occupancies, making it particularly attractive for this use case. Based on the results, it can be inferred that the ICR method outperformed the unmodified method, with the greatest increase in detectability occurring around test 57.

Table 4
Test results when considering the minimization of memory occupancy

#	W	P	$TPI\%$	$G\%$	Rm	Rm_I	Rm_T	$Rm_T\%$
1	24, 9	16, 8	81	3	279	26	305	3.3
2	24, 9	16, 9	76	7	279	194	473	5.1
3	20, 9	16, 10	83	47	68	524	592	6.4
4	27, 10	16, 8	77	0	645	0	645	7
5	20, 10	16, 8	82	0	700	0	700	7.6
6	29, 10	16, 9	77	1	693	10	703	7.6
7	20, 10	16, 9	80	1	700	46	746	8.1
8	32, 9	16, 8	86	0	684	72	756	8.2
9	16, 8	16, 10	87	67	24	762	786	8.5
10	16, 10	16, 8	84	0	762	24	786	8.5

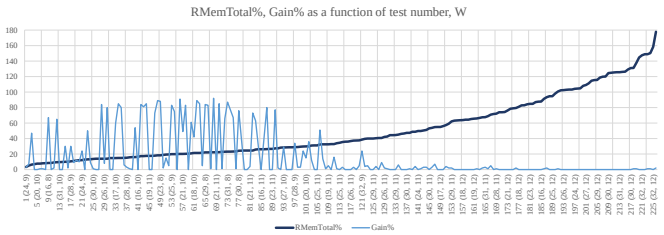

Fig. 8. Receptor memory occupancy relative to program size [%] and intercellular detection rate gains [%] as a function of test number and W – the criterion of memory occupancy minimization

5.4. The criterion of maximizing program speed

The criterion for maximizing speed is to achieve the fastest possible receptor generation (lowest Gt) while maintaining a detection rate of at least 75%. It is assumed that the execution time of the generation algorithm should not exceed 2 seconds. The results are shown in Table 5 and are sorted first by the lowest value of Gt.

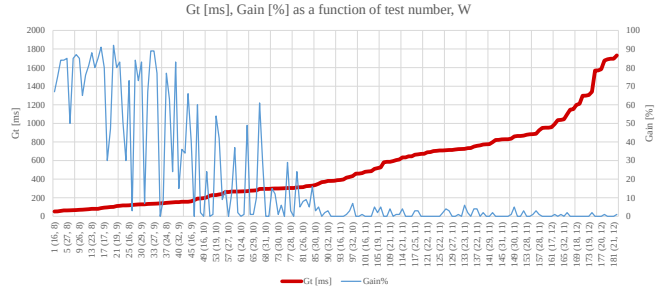

Fig. 9. Total generation time [ms] and intercellular detection rate gains [%] as a function of test number and W – the criterion of maximization of program speed

Table 5
Test results when considering the maximization of program speed

#	W	P	$TPI\%$	$G\%$	$Rm_T\%$	TGt	RGt	Gt
1	16, 8	16, 10	87	67	8.5	22	3	52
2	25, 8	16, 11	75	75	19.9	24	1	53
3	17, 8	16, 11	84	84	14	16	0	58
4	29, 8	16, 11	84	84	22.2	29	1	63
5	27, 8	16, 11	85	85	21.5	27	1	64
6	32, 8	16, 10	84	50	13.2	47	3	65
7	19, 8	16, 11	85	85	15	20	0	66
8	31, 8	16, 11	87	87	23.2	31	0	67
9	26, 8	16, 11	85	85	22.4	28	4	70
10	24, 8	16, 10	76	65	9.5	37	4	71

Figure 9 shows the test results for the speed maximization criterion. It can be seen that the ICR method produced up to an 87 percentage point increase in detection rate, with generation times as low as 52 ms. This confirms that the ICR method is suitable for use with this criterion.

5.5. Comparison with other solutions

The primary evaluation of the proposed IDS in this work is carried out on compiled program binaries, where controlled anomalies are injected into an executable and detection rates, memory occupancy and program speed are measured. This setting reflects the intended deployment scenario: host-based, static analysis of program files. In addition to these domain-specific experiments, we also wished to compare the ICR-enhanced NSA against several well-known NSA variants under identical conditions, focusing on algorithmic properties such as false positives and receptor counts.

There is currently no widely adopted benchmark dataset for static, binary-level program anomaly detection. Following common practice in the AIS and NSA literature, the classic Iris flower measurement dataset was used as a pattern-recognition benchmark for this comparative study, treating each species in turn as the "self" class. The experiments in this subsection should be interpreted as algorithmic comparisons, as they illustrate how the ICR modification behaves relative to other NSA variants on a standard classification task. The dataset contains 50 records for each of the three species of Iris flower: Iris setosa, Iris versicolor, and Iris virginica. Each record contains four values of flower petal measurements and information about its species. This set was chosen to compare the performance of IDS with other solutions. Table 6 shows sample records from the dataset. To ensure compatibility with IDS, this collection was stored as a sequence of 32-bit floating-point numbers.

The algorithms chosen for performance comparison were NSA-R (where receptors are selected randomly), NSA-T (where receptors are selected using templates), MILA (multilevel immune learning algorithm), V-detector, FB-NSA, and the ICR algorithm, which includes our proposed modification. The number of receptors to be generated for the NSA-R and MILA algorithms was set at 1000, with the MILA algorithm containing 1000 T-type receptors and 1000 B-type receptor groups due to its design. It was assumed that the parameter m (activation threshold) for the algorithms was 6, and the parameter l (receptor length) was, in the case of the ICR method, tested in the range $\{16, 17, ..., 28\}$. The parameter l_I (length of intercellular receptors) for testing was set to a value of 16, and the value of m_I (activation threshold of intercellular receptors) was tested over the range $\{2, 3, ..., 8\}$. The parameters tested were $TP\%$ (average true positive ratio), $FP\%$ (average false positive ratio), and Rn, the average number of receptors in the **R** set (and in the case of the ICR algorithm, the union of the **R** and $\mathbf{R_I}$ sets). The ICR algorithm was trained for each flower species separately. The algorithm was first trained using all records of the Iris setosa species, and then tested on 100 random records from the other two species. The results were recorded, and then the algorithm was retrained using only 50% of the records of the Iris setosa species. The 100 tests were run again, the results were recorded, and then the whole procedure was then repeated for the Iris versicolor species, and so on until all three species were tested. The same assumptions were made for the other algorithms. Table 7 shows the performance of the algorithms for each flower species.

Table 6
Sample records from the Iris flower dataset

#	Sepal len.	Sepal width	Petal len.	Petal width	Species
1	5.1	3.5	1.4	0.2	I. setosa
2	4.9	3.0	1.4	0.2	I. setosa
3	4.7	3.2	1.3	0.2	I. setosa
4	4.6	3.1	1.5	0.2	I. setosa
5	5.0	3.6	1.4	0.3	I. setosa
...					

Table 7
Comparison with other solutions for Iris setosa

Dataset	Algorithm	$TP\%$	$FP\%$	Rn
Iris setosa 100%	NSA-R	100	0	1000
	NSA-T	87.18	0	77.95
	MILA	95.16	0	1000
	V-detector	99.98	0	20
	FB-NSA	100	0	83
	ICR	94.84	0	111.3
Iris setosa 50%	NSA-R	100	11.18	1000
	NSA-T	97.44	6.1	91.46
	MILA	94.02	8.42	1000
	V-detector	99.97	1.32	16.44
	FB-NSA	100	2.76	76.97
	ICR	98.17	6.7	124.81

For the full training set of Iris setosa, it can be observed that the NSA-R and FB-NSA algorithms achieved 100% detection. However, in the case of NSA-R, this result was achieved by placing as many as 1 000 receptors in memory. The ICR, MILA, and V-detector algorithms achieved detection rates of over 90%. It should be noted, however, that the MILA algorithm generated 1 000 receptors but showed lower detection than NSA-R, making it the least viable for this dataset and configuration. The V-detector algorithm achieved a detection rate of 99.98% with 20 generated receptors. The ICR method was able to detect anomalies 94.84% of the time with an average of 111.3 generated receptors. For this dataset, it ranks fifth in detection and fourth in terms of receptor count. However, the results are comparable. It is noteworthy that none of the methods falsely detected any abnormalities. This is due to the complete training of the negative selection algorithm, which is only possible with full knowledge of the self-pattern set. The situation changed when the training data was limited to 50% of the Iris setosa records. The detection rates of all methods except MILA increased. However, due to the limited training data, not all records were classified as self fragments. This led to false classification of Iris setosa records as nonself. The FP% was 11.18 for the NSA-R algorithm. The best performer in this test was the V-detector, which achieved a low FP% of 1.32. The ICR algorithm, despite achieving a 98.17% detection rate in this test, showed a rather high receptor memory occupancy compared to other methods.

Table 8
Comparison with other solutions for Iris versicolor

Dataset	Algorithm	TP%	FP%	Rn
Iris versicolor 100%	NSA-R	95.67	0	1000
	NSA-T	82.05	0	76.82
	MILA	84.37	0	1000
	V-detector	85.95	0	153.24
	FB-NSA	92	0	299
	ICR	88.27	0	110.24
Iris versicolor 50%	NSA-R	96	22.2	1000
	NSA-T	93.19	15.5	95.72
	MILA	84.46	19.6	1000
	V-detector	88.3	8.42	110.08
	FB-NSA	95.17	10.52	282.61
	ICR	97.9	16.1	129.14

When the training set was the full Iris versicolor set (Table 8), the detection rates of all algorithms decreased slightly compared to the previous species. The smallest decrease in detection rate occurred with the NSA-R method (4.33 percentage points). In comparison, the detections of the NSA-T, MILA, V-detector, FB-NSA, and ICR algorithms decreased by 5.13, 10.79, 14.03, 8 and 6.57 percentage points, respectively. The ICR algorithm achieved a detection rate better than that of the NSA-T, MILA and V-detector algorithms, but generated fewer receptors only compared to the MILA and V-detector algorithms. The highest detectability was achieved by the FB-NSA algorithm, which reached 92% detection. Its execution, however, resulted in the generation of 299 receptors. More efficient in this regard was the proposed ICR algorithm, which created only 110.24 receptors on average, achieving a detection rate just 3.73 percentage points lower. With 50% of the Iris versicolor data, both false positives and detection rates increased, as seen with the previous species. The algorithm that achieved the best detection in this test was the proposed ICR algorithm. Tests showed that this approach resulted in the detection of 97.9% of anomalies, which was almost 2 percentage points higher than the second-best NSA-R algorithm. This improvement is notable given that the average number of receptors was 129.14, and in the case of NSA-R, 1000. The NSA-T and V-detector algorithms achieved lower receptor numbers, but also lower detection rates. The MILA method proved to be one of the least effective in this test, achieving a detection rate of 84.46% with 1000 receptors.

Table 9
Comparison with other solutions for Iris virginica

Dataset	Algorithm	TP%	FP%	Rn
Iris virginica 100%	NSA-R	90.38	0	1000
	NSA-T	79.49	0	80.15
	MILA	75.75	0	1000
	V-detector	81.87	0	218.36
	FB-NSA	94	0	207
	ICR	94.1	0	115.32
Iris virginica 50%	NSA-R	97.18	33.26	1000
	NSA-T	90.91	25.39	97.72
	MILA	88.96	24.98	1000
	V-detector	93.58	13.18	108.12
	FB-NSA	95.78	16.64	194.37
	ICR	96.08	28	132.88

For the full set of Iris virginica (Table 9), it is noteworthy that among those tested, the ICR algorithm again achieved the best detection rate. The intercellular receptors used in this algorithm helped increase the detection rate relative to the NSA-T method by 14.61 percentage points, yielding an average detection rate of 94.1 TP%. It is worth noting that the FB-NSA algorithm had a comparable detection rate of 94% but required nearly twice the receptor memory. In this case, the ICR algorithm generated an average of 115.32 receptors per test, while the FB-NSA algorithm generated 207. The other methods achieved TP% values ranging from 75.75 to 90.38. The only method that generated fewer receptors than ICR was the NSA-T method, which used an average of eighty receptors. With only 50% of the Iris virginica data used for training, high FP% values were observed, reaching up to 33.26 in some algorithms. The algorithm that reached

this ceiling was NSA-R. The reduction of self fragment data had the least impact on the V-detector and FB-NSA algorithms, which achieved $FP\%$ values of 13.18 and 16.64, respectively. The $TP\%$ rate for FB-NSA was 95.78, while the ICR algorithm achieved a $TP\%$ 0.3 points higher. The FB-NSA method had a lower false alarm rate than ICR, but this came at the cost of significantly more receptors in memory.

5.6. Evaluation with VirusShare malware fragments

To complement the experiments on synthetic anomalies in test.exe and the algorithmic comparison using the Iris dataset, and to gauge the performance of the IDS using a domain-relevant dataset, additional tests were performed using real malware code fragments taken from the VirusShare repository. VirusShare is a large, publicly available collection of malware samples. For consistency with the rest of the experiments, 32-bit Windows PE files whose structure is compatible with the monitored program described in Table 2 were used.

Table 10
Test results for 10 VirusShare malware samples

Sample #	$TPI\%$
1	95
2	94
3	95
4	95
5	95
6	95
7	95
8	96
9	94
10	95

For these experiments, the IDS implementation and test environment from Sections 5.1 – 5.4 were used without any changes. The parameter configuration that maximized detection rate in Table 3, namely $W(20, 10)$ for standard receptors and $P(16, 10)$ for intercellular receptors, was selected for testing. This configuration was used to generate the receptor sets $\mathbf{R}$ and $\mathbf{R_I}$ from the original, unmodified test.exe file, as in the baseline experiments.

A set of 10 VirusShare samples was then chosen, each being a 32-bit Windows PE executable. From the code section of each sample, 100 random 4-byte fragments were extracted, skipping any regions that could not be parsed as code. Each such fragment was treated as a candidate anomaly. Following the process in Section 5.1, these 4-byte malware-derived fragments were injected into test.exe at random locations, one fragment at a time, restoring the original executable after each run. In total, 1000 detection attempts were performed with malware-derived anomalies, logging whether the IDS raised a true positive for each injected fragment.

The results of this experiment are presented in Table 10. The ICR-enhanced NSA achieved an overall average detection rate of 94.9% on malware-derived anomalies, with per-sample detection rates ranging from 94% to 96%. The results indicate that the detection rates provided by intercellular receptors are not limited to synthetic, uniformly random modifications, but also carries over to byte patterns taken from real malware samples. At the same time, the overall computational cost and memory footprint remained comparable to those observed in the synthetic anomaly experiments, since the receptor sets $\mathbf{R}$ and $\mathbf{R_I}$ were generated from the same monitored program and with the same parameter values.

This experiment focuses on static byte-level modifications embedded into a single host program. A more comprehensive evaluation, including packed and obfuscated samples and different host binaries is a potential direction for future work.

5.7. Practical considerations and limitations

The proposed IDS is tailored for real-world settings that prioritize lightweight and efficient anomaly detection. With low memory requirements and fast receptor generation, it fits comfortably in contexts like embedded systems, operating-system-level file monitoring, and other environments where minimizing overhead is critical. At this stage, however, the system is confined to static, binary-level scanning of program files; it does not yet accommodate dynamic detection or runtime analysis. Furthermore, it is not engineered for network-based intrusion detection, which limits its role in broader IDS architectures. Advancing the method to cover runtime and network threats thus becomes a key direction for future research.

Scalability. The experimental configuration in Tables 3, 4, 5, and 10 uses a 9216-byte executable and shows that receptor generation times remain below 2 seconds even for the most demanding parameter settings, with many useful configurations completing within tens of milliseconds. Algorithmically, receptor generation and anomaly checking scale approximately linearly with both the number of monitored cells and the number of receptors in $\mathbf{R} \cup \mathbf{R_I}$, because each receptor inspects a fixed-size window over the program. This implies that scanning larger binaries or directories will increase both computation time and memory footprint proportionally to receptor count. In practice, this cost can be controlled by tuning l, m, l_I, m_I to reduce the required number of receptors, caching receptors for frequently scanned binaries, and parallelizing both generation and detection across files. A full evaluation of large programs and long-running monitoring workloads is left to future research, but the results reported in this paper indicate that the ICR modification does not introduce any additional bottlenecks beyond those already present in NSA-based IDS.

False positives and false negatives. The mathematical model in Section 2 explicitly distinguishes between true/false positives and negatives, and the receptor generation procedure enforces the self/nonself constraint in (1), which eliminates any candidate that activates on the self set. In the binary-level experiments on test.exe this means that, under the assumption of a complete self set for that program, the system does not generate false alarms on the monitored file; instead, we study how often in-

jected anomalies are detected (TP). In realistic deployments the self set is rarely complete, and (1) can only be enforced with respect to a limited training sample. The algorithmic comparisons (Tables 7, 8, 9) illustrate this regime: when the training set is reduced to 50% of the available self data, all algorithms – including the ICR-enhanced NSA – start to exhibit non-zero FP%. These results highlight the usual IDS trade-off: more aggressive receptor coverage and lower activation thresholds increase the detection rate, but they can also increase the false-positive rate when the model of normal behavior is incomplete. The ICR mechanism fits into this trade-off by expanding coverage between cells; it can improve TP but may require careful tuning of parameters m and m_I to keep FP rates acceptable.

Adversarial evasion. Like most signature- and pattern-based IDS mechanisms, the proposed system assumes that attacks manifest as localized binary modifications that differ sufficiently from the learned self patterns. An adaptive adversary may attempt to evade detection by distributing modifications across more than two adjacent cells so that neither $\mathbf{R}$ nor $\mathbf{R_I}$ sees an anomalous subsequence of length m and m_I, inserting code sequences that deliberately mimic frequent self fragments, or relocating malicious logic to regions of the binary that are not monitored. The introduction of ICRs makes simple "between-cell" modifications harder to hide, but it does not make the system robust against such targeted evasion strategies. Mitigating these attacks will likely require combining static ICR-based detection with runtime behavior analysis, or integrating additional features beyond monitoring raw bytes.

6. CONCLUDING REMARKS

In this paper, we proposed an intrusion detection system (IDS) based on the negative selection algorithm (NSA) from the domain of artificial immune systems (AIS). The proposed intercellular modification of the NSA (ICR) was described in detail and implemented practically within an operating system environment. Extensive testing demonstrated that the IDS is an effective method for detecting program-level anomalies, particularly when using activation threshold values of $m \leq 10$. Depending on the configuration, the ICR-enhanced NSA achieved detection improvements of up to 42 percentage points, with an average gain of 8.1 percentage points over the standard method, while also retaining low memory usage and maintaining low generation time. These characteristics make the system especially well-suited for applications requiring minimal overhead, such as embedded systems or OS-level file monitoring. Comparisons with other state-of-the-art algorithms showed that the proposed approach can produce comparable or superior results, often with fewer receptors and lower resource consumption. Future research may include calculating probabilistic bounds for all 340 tested configurations to better understand detection coverage. We also plan to adapt the algorithms for potential use with network-level intrusion scenarios, extending the evaluation to benchmarks such as NSL-KDD and CICIDS2017. This will allow us to assess the generalizability of the ICR-enhanced NSA to broader IDS domains.

REFERENCES

[1] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, "A sense of self for unix processes," in *Proc. 1996 IEEE Symposium on Security and Privacy*, 1996, pp. 120–128, doi: 10.1109/SECPRI.1996.502675.

[2] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a computer immune system," in *Proc.e 1997 Workshop on New Security Paradigms, NSPW 1997*, 1997, pp. 75–82.

[3] A. Forrest, S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *Proc. 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, 1994, pp. 202–212, doi: 10.1109/RISP.1994.296580.

[4] J. Kephart, "A biologically inspired immune system for computers," in *Fourth International Workshop on Synthesis, Simulation of Living Systems, Artificial Life IV*, 1994, pp. 130–139.

[5] D. Dasgupta, "Immunity-based intrusion detection system: A general framework," in *Proc. 22nd National Information Systems Security Conference*, 1999, pp. 147–160.

[6] D. Dasgupta and F. Gonzalez, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Trans. Evol. Comput.*, vol. 6, no. 3, pp. 281–291, 2002, doi: 10.1109/TEVC.2002.1011541.

[7] P.S. Andrews and J. Timmis, *Tunable Detectors for Artificial Immune Systems: From Model to Algorithm*. New York, NY: Springer New York, 2010, pp. 103–127, doi: 10.1007/978-1-4419-0540-6_9.

[8] T.S. Sobh, "A cooperative immunological approach for detecting network anomaly," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 1275–1283, 2011, doi: 10.1016/j.asoc.2010.03.004.

[9] D. Wang, F. Zhang, and L. Xi, "Evolving boundary detector for anomaly detection," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2412–2420, 2011, doi: 10.1016/j.eswa.2010.08.030.

[10] S. T. Powers and H. J., "A hybrid artificial immune system and self organising map for network intrusion detection," *Inf. Sci.*, vol. 178, no. 15, pp. 3024–3042, 2008, doi: 10.1016/j.ins.2007.11.028.

[11] G.-Y. Li and T. Guo, "Receptor editing-inspired negative selection algorithm," in *2010 International Conference on Machine Learning and Cybernetics*, vol. 6, 2010, pp. 3117–3122, doi: 10.1109/ICMLC.2010.5580727.

[12] C. Laurentys, G. Ronacher, R. Palhares, and W. Caminhas, "Design of an artificial immune system for fault detection: A negative selection approach," *Expert Syst. Appl.*, vol. 37, no. 7, pp. 5507–5513, 2010, doi: 10.1016/j.eswa.2010.02.004.

[13] R. Fanelli, "A hybrid model for immune inspired network intrusion detection," in *Artificial Immune Systems*, P. Bentley, D. Lee, and S. Jung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 107–118.

[14] P. Mostardinha, B.F. Faria, A. Zúquete, and F. Vistulo de Abreu, "A negative selection approach to intrusion detection," in *Artificial Immune Systems*, C. A. Coello Coello, J. Greensmith, N. Krasnogor, P. Liò, G. Nicosia, and M. Pavone, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 178–190.

[15] Z. Ji and D. Dasgupta, "Real-valued negative selection algorithm with variable-sized detectors," in *Genetic and Evolutionary Computation – GECCO 2004*, K. Deb, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 287–298.

[16] D. Dasgupta, S. Yu, and N.S. Majumdar, "Mila — multilevel immune learning algorithm," in *Genetic and Evolutionary Com-*

putation — GECCO 2003, E. Cantú-Paz *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 183–194.

[17] D. Li, S. Liu, and H. Zhang, "Negative selection algorithm with constant detectors for anomaly detection," *Appl. Soft Comput.*, vol. 36, no. C, p. 618–632, Nov. 2015, doi: 10.1016/j.asoc.2015.08.011.

[18] B. Bejoy, G. Raju, D. Swain, B. Acharya, and Y.-C. Hu, "A generic cyber immune framework for anomaly detection using artificial immune systems," *Appl. Soft Comput.*, vol. 130, p. 109680, 2022, doi: 10.1016/j.asoc.2022.109680.

[19] Y.J. Kim, W. Nam, and J. Lee, "Multiclass anomaly detection for unsupervised and semi-supervised data based on a combination of negative selection and clonal selection algorithms," *Appl. Soft Comput.*, vol. 122, p. 108838, 2022, doi: 10.1016/j.asoc.2022.108838.

[20] J. He, W. Chen, T. Li, B. Li, Y. Zhu, and M. Huang, "Hd-nsa: A real-valued negative selection algorithm based on hierarchy division," *Appl. Soft Comput.*, vol. 112, p. 107726, 2021, doi: 10.1016/j.asoc.2021.107726.

[21] H. Alrubayyi, G. Goteng, M. Jaber, and J. Kelly, "A novel negative and positive selection algorithm to detect unknown malware in the iot," in *IEEE INFOCOM 2021 – IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–6, doi: 10.1109/INFOCOMWKSHPS51825.2021.9484483.

[22] K.D. Gupta and D. Dasgupta, "Negative selection algorithm research and applications in the last decade: A review," *IEEE Trans. Artif. Intell.*, vol. 3, no. 2, p. 110–128, Apr. 2022, doi: 10.1109/tai.2021.3114661.

[23] K. Wawryn and P. Widulinski, "A human immunity inspired algorithm to detect infections in a computer program," in *2019 MIXDES - 26th International Conference "Mixed Design of Integrated Circuits and Systems"*, 2019, pp. 381–385, doi: 10.23919/MIXDES.2019.8787193.

[24] P. Widulinski and K. Wawryn, "A human immunity inspired intrusion detection system to search for infections in an operating system," in *2020 27th International Conference on Mixed Design of Integrated Circuits and System (MIXDES)*, 2020, pp. 187–191, doi: 10.23919/MIXDES49814.2020.9155771.

[25] P. Widulinski and K. Wawryn, "A study of detection probabilities and real-world testing of a human immunity inspired intrusion detection system," in *2021 28th International Conference on Mixed Design of Integrated Circuits and System*, 2021, pp. 261–264, doi: 10.23919/MIXDES52406.2021.9497536.

[26] P. Widulinski and K. Wawryn, "Parameter efficiency testing for an intrusion detection system inspired by the human immune system," in *2022 29th International Conference on Mixed Design of Integrated Circuits and System (MIXDES)*, 2022, pp. 208–212, doi: 10.23919/MIXDES55591.2022.9838210.

[27] K. Wawryn and P. Widuliński, "Detection of anomalies in compiled computer program files inspired by immune mechanisms using a template method," *J. Comput. Virol Hacking Tech.*, vol. 17, pp. 47–59, 2021, doi: 10.1007/s11416-020-00364-w.

[28] Y. Chen, H. Wang, W. Li, and J. Luo, "Cost reassignment for improving security of adaptive steganography using an artificial immune system," *IEEE Signal Process Lett.*, vol. 29, pp. 1564–1568, 2022, doi: 10.1109/LSP.2022.3188174.

[29] W. Li, H. Wang, Y. Chen, S. M. Abdullahi, and J. Luo, "Constructing immunized stego-image for secure steganography via artificial immune system," *IEEE Trans. Multimedia*, vol. 25, pp. 8320–8333, 2023, doi: 10.1109/TMM.2023.3234812.

[30] N. Elisa, L. Yang, F. Chao, N. Naik, and T. Boongoen, "A secure and privacy-preserving e-government framework using blockchain and artificial immunity," *IEEE Access*, vol. 11, pp. 8773–8789, 2023, doi: 10.1109/ACCESS.2023.3239814.

[31] H. Xuan, W. Li, and B. Li, "An artificial immune differential evolution algorithm for scheduling a distributed heterogeneous flexible flowshop," *Appl. Soft Comput.*, vol. 145, p. 110563, 2023, doi: 10.1016/j.asoc.2023.110563.

[32] S. T. Wierzchon, *Sztuczne systemy immunologiczne. Teoria i zastosowania*. Warszawa: Akademicka Oficyna Wydawnicza EXIT, 2001, (in Polish).