Zeszyty Naukowe Wydziału
Elektroniki i Informatyki
Politechniki Koszalińskiej

ISSN: 1897-7421
Nr 14, Czerwiec 2019
Str: 23–41

Patryk Widuliński*
Krzysztof Wawryn
Wydział Elektroniki i Informatyki
Politechnika Koszalińska
* patryk.widulinski@tu.koszalin.pl

Detekcja anomalii w plikach za pomocą wybranych algorytmów inspirowanych mechanizmami immunologicznymi

Słowa kluczowe: systemy wykrywania intruzów, sztuczne systemy immunologiczne, wirusy, szkodliwe oprogramowanie, algorytm negatywnej selekcji, generacja receptorów, anomalia, wykrywanie anomalii

1. Wstęp

Od ostatnich kilkudziesięciu lat ochrona przed atakami wirusowymi jest ważnym zadaniem projektantów oprogramowania zabezpieczającego. Wskutek tego na całym świecie postępuje rozwój systemów wykrywania intruzów (SWI). Systemy te używane są na całym świecie do zabezpieczania przed atakami sieciowymi na poziomie zarówno lokalnym, jak i globalnym. Problemami, które między innymi napotykają projektanci oprogramowania zabezpieczającego są wykrywanie wcześniej nieznanych ataków i rosnąca złożoność oprogramowania komputerowego. Nakłonieni do poszukiwania nowatorskich rozwiązań tych problemów, projektanci dostrzegli analogie między zadaniami realizowanymi przez systemy wykrywania intruzów i ludzki układ odpornościowy. Przysposobienie mechanizmów ludzkiego układu odpornościowego pozwala na rozwiązanie wymienionych wyzwań. W literaturze pojawiło się dużo podejść do tematu systemów wykrywania intruzów bazujących na sztucznych systemach immunologicznych. Większość z nich korzysta z systemów wykrywania anomalii i algorytmach selekcji negatywnej w celu wykrywania infekcji w oprogramowaniu komputerowym. Metoda negatywnej selekcji oparta jest na rozpoznawaniu wzorców własnych i obcych i prowadzi do wykrywania anomalii w oprogramowaniu. Podstawy mechanizmów

systemów wykrywania intruzów opartych o sztuczne systemy immunologiczne opisane są w [1-5]. Inicjują one bardziej skomplikowane systemy zaprezentowane w [6-13].

W artykule zaprezentowano system wykrywania intruzów w postaci nieprawidłowości (infekcji) w treści skompilowanych programów komputerowych. Mechanizm wykrywania oparty jest na algorytmie negatywnej selekcji. Składa się z dwóch zadań: generacji receptorów i wykrywania anomalii. Sekcje kodu w programach reprezentowane są przez ciągi binarne. Generacja receptorów realizowana jest na dwa sposoby: losowo i korzystając z algorytmu negatywnej selekcji do podziału zbioru szablonów na własne i obce. Szablony własne są odrzucane, tak, by infekcje wykrywały tylko szablony obce, które stają się receptorami. Anomalie w programie wykrywane są przy pomocy wygenerowanych receptorów. Program jest zainfekowany, jeżeli między jego fragmentem a jednym z receptorów występuje określona liczba bitów pasujących.

Artykuł zorganizowany jest w następujący sposób. Proponowany system wykrywania intruzów omówiono w rozdziale 2. Metodę losową omówiono w rozdziale 3. Metodę szablonów omówiono w rozdziale 4. Obie metody przebadano eksperymentalnie i przedstawiono wyniki w rozdziale 5. Następnie dokonano analizy porównawczej obu metod w rozdziale 6, a w rozdziale 7 przedstawiono wnioski.

2. System wykrywania intruzów

Proponowany przez nas system wykrywania intruzów (SWI) może być wykorzystywany do wykrywania nieregularności w programach komputerowych w ich skompilowanej formie.

SWI monitoruje obszar w systemie operacyjnym podatny na infekcje. W przypadku systemu Microsoft Windows mógłby to być przykładowo katalog C:\Windows. Monitorowany obszar zawiera programy, które muszą być chronione przez SWI. W celu umożliwienia poprawnego działania SWI, programy w tym obszarze muszą inicjalnie istnieć w ich niezmienionej (niezainfekowanej) formie. Programy w tym obszarze muszą również być prawidłowymi plikami wykonywalnymi w formacie Win32 PE (Portable Executable). Typowy, prawidłowy plik PE zawiera sekcje które mają określone właściwości informujące o ich przeznaczeniu (sekcje kodu, zainicjalizowanych danych, niezainicjalizowanych danych) i uprawnieniach odczytu, zapisu i wykonania względem procesora głównego (CPU).

Proponowany system zawiera główny blok sterujący, który nadzoruje działanie bloków generacji receptorów i wykrywania anomalii. System skanuje pliki PE w monitorowanym obszarze w poszukiwaniu sekcji kodu, a następnie generuje specjalne ciągi binarne nazwane receptorami dla każdego pliku. Wygenerowane

receptory przechowywane są w osobnym zbiorze i wykorzystywane są do wykrywania anomalii w plikach w obszarze monitorowanym.

W SWI zaimplementowano dwie różne metody do generacji receptorów: metodę losową i metodę szablonów. Użytkownik systemu decyduje z jakiej metody ma korzystać system.

3. Metoda losowa

3.1. Generacja receptorów

W celu wykrywania nieregularności (anomalii) w kodzie programu, SWI musi skonstruować zbiór receptorów.

Receptorami nazywamy ciągi binarne o długości l. Jeśli ciągi te zostaną użyte w określonych wzorach, zyskują one zdolność do rozpoznawania obcych struktur w kodzie. Schemat blokowy generacji losowej receptorów przedstawiono na Rys. 1. Liczba receptorów do wygenerowania przez system określona jest przez użytkownika i oznaczona jest jako R_{max}. Maksymalna liczba receptorów, które mogą być wygenerowane przez system dla pojedynczego pliku jest zdefiniowana następującym równaniem:

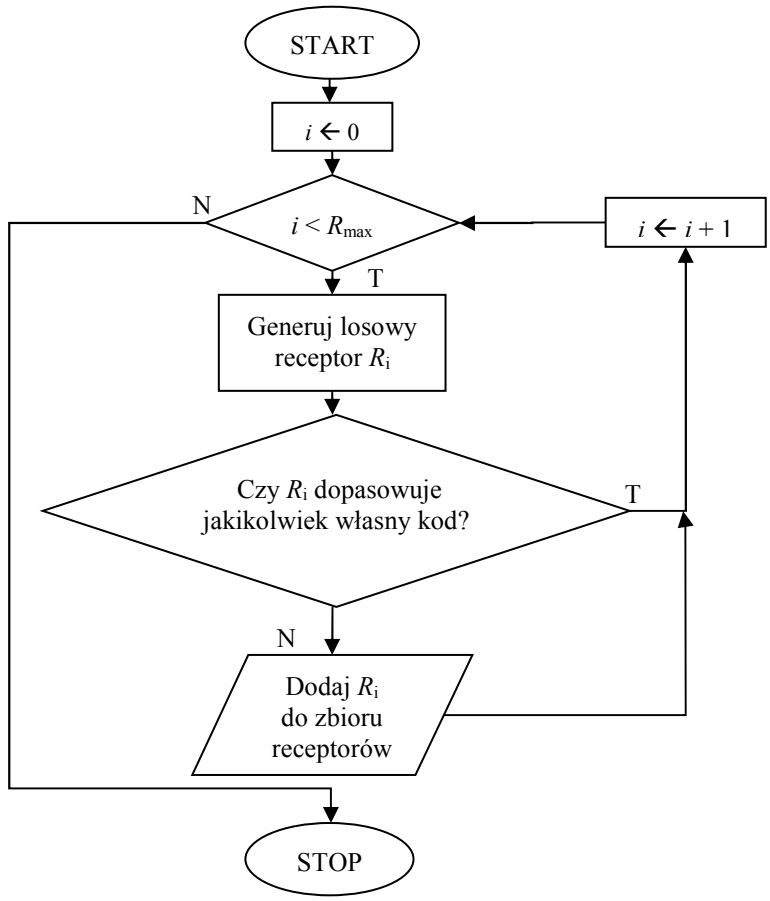
$$max(R_{max}) = 2^l, \tag{1}$$

gdzie l oznacza długość receptora w bitach.

Metoda losowa zakłada losową generację receptorów. W algorytmie tym wykorzystano generator 8-bitowych liczb pseudolosowych do generacji receptorów o rozmiarze l. Generator wytwarza pseudolosowe liczby 8-bitowe z zakresu [0; 255]. Liczba 8-bitowych fragmentów receptora do losowej generacji oznaczona jest przez f i określona równaniem 2:

$$f = ceil(div(l, 8)), \tag{2}$$

gdzie l jest długością receptora w bitach, $div(a, b)$ jest wynikiem dzielenia a przez b i $ceil(a)$ jest funkcją zaokrąglającą liczbę a w górę. Po wygenerowaniu 8-bitowych fragmentów są one łączone ze sobą w jeden ciąg binarny na zasadzie konkatenacji, tworząc losowy receptor.



Rys. 1. Schemat blokowy algorytmu losowej generacji receptorów

Jeśli l nie jest podzielne przez 8, w receptorze obecne są bity zbędne. Zbędne bity przechowywane są w pamięci, ale nie są dalej używane w algorytmach generacji i detekcji. Przykłady losowej generacji receptorów pokazane są na Rys. 2 i Rys. 3.

01001011	**11011001**	**00000101**	**1110**1111
4B	**D9**	**05**	**E**F

Rys. 2. Przykład losowej generacji receptora dla $l = 28, f = 4$ (wyżej: reprezentacja binarna, niżej: reprezentacja szesnastkowa), bity niepogrubione są zbędne

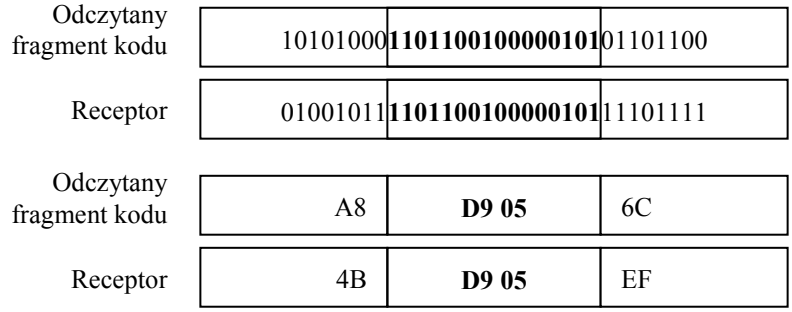
11100001	**10011010**
E1	**9A**

Rys. 3. Przykład losowej generacji receptora dla $l = 16$, $f = 2$ (wyżej: reprezentacja binarna, niżej reprezentacja szesnastkowa), brak bitów zbędnych

Na powyższych rysunkach przedstawiono dwa przykłady losowej generacji receptorów. Na Rys. 2., dla $l = 28$ generator liczb pseudolosowych wygenerował 4 bajty: $(4B)_{16}$, $(D9)_{16}$, $(05)_{16}$ i $(EF)_{16}$. Bajty te zostały połączone ze sobą następnie w procesie konkatenacji, co spowodowało utworzenie 32-bitowego receptora $(4BD905EF)_{16}$. Jednakże, ponieważ $l = 28$, cztery bity tego receptora nie będą dalej wykorzystane przez zaproponowane algorytmy. W przykładzie na Rys. 3. bity zbędne są nieobecne, ponieważ liczba $l = 16$ jest podzielna przez 8, toteż liczby 8-bitowe $(E1)_{16}$ i $(9A)_{16}$ utworzyły receptor $(E19A)_{16}$ i będzie on w całości dalej wykorzystany – jeżeli pomyślnie przejdzie testy dopasowania struktur własnych.

Cechą charakterystyczną podejścia inspirowanego sztucznymi systemami immunologicznymi, wykorzystującego receptory jest obecność progu aktywacji, oznaczonego m. Próg aktywacji jest minimalną liczbą kolejnych bitów, które muszą być dopasowane między receptorem a odczytanym fragmentem programu by receptor ten można było uznać za dopasowany (lub „aktywowany"). Jeżeli liczba dopasowanych kolejno bitów między receptorem a odczytanym fragmentem kodu jest mniejsza niż m, to receptor ten nie jest uznawany za dopasowany/aktywowany.

Ponieważ celem receptora jest wykrywanie struktur obcych, niezbędne jest sprawdzenie każdego losowo wygenerowanego receptora pod kątem wykrywalności kodu własnego. W tym celu algorytm otwiera plik PE przez interfejs systemu plików i lokalizuje informacje na temat sekcji kodu. Następnie system porównuje nowo wygenerowany receptor ze wszystkimi fragmentami kodu w sekcjach kodu w programie PE. By osiągnąć poprawne porównanie, algorytm odczytuje dane pliku fragmentami o długości l (takiej samej jak długość receptora), zaczynając od początku każdej sekcji kodu.



Rys. 4. Przykład receptora dopasowanego do odczytanego fragmentu programu dla $l = 32$, $m = 16$, receptor został dopasowany z przesunięciem okna $k = 8$

Kolejne m bitów nazywane jest oknem. Obecna pozycja okna względem receptora nazywana jest przesunięciem okna i oznaczona jest przez k. Jeżeli nowo wygenerowany receptor zostanie dopasowany w jakimkolwiek miejscu w programie (dopasowane zostanie co najmniej m bitów między receptorem a programem, z dowolnym przesunięciem okna k), to jest on odrzucany i nie może być wykorzystany do wykrywania anomalii, ponieważ wykrywa on również kod własny. W przypadku gdy receptor nigdy nie dopasowuje struktur własnych dodawany jest on do finalnego zbioru receptorów. Ostateczna liczba receptorów dodana do zbioru oznaczona jest przez R_n.

3.2. Wykrywanie anomalii

Proponowany system wykrywa anomalie w kodzie programów w obszarze monitorowanym przy użyciu receptorów wygenerowanych na etapie, gdy pliki były w ich zaufanym, inicjalnym stanie.

Zanim algorytm rozpocznie skanowanie programów PE w celu poszukiwania anomalii, zliczane są receptory obecne w zbiorze receptorów. Liczba ta oznaczana jest jako R_n. Dla każdego receptora obecnego w zbiorze, algorytm dokonuje odczytu fragmentów sekcji kodu o rozmiarze l i porównuje odczytane próbki z receptorem.

Jeżeli dopasowane zostanie m kolejnych bitów, receptor jest aktywowany i anomalia zostaje w ten sposób wykryta. Informacja o wykryciu anomalii jest przekazywana następnie użytkownikowi, wraz z informacją o lokalizacji wystąpienia anomalii.

4. Metoda szablonów

4.1. Generacja receptorów

Generacja receptorów metodą szablonów w proponowanym systemie opiera się na zmodyfikowanej metodzie opartej o szablony wprowadzonej w [14] i rozwiniętej w [15].

Szablonem nazywamy ciąg binarny zawierający bity istotne i nieistotne. Bity istotne w szablonach mają ustaloną wartość, a bity nieistotne oznaczane są przez gwiazdkę „*”.

W pierwszym kroku konstruowana jest tablica **T**, składająca się z szablonów o długości l bitów. Dla określonego progu aktywacji m w szablonach występuje dokładnie m ustalonych bitów. Pozostałe $l – m$ bitów w szablonie jest nieistotne.

Liczba wszystkich szablonów określona jest następującym wzorem:

$$L_s = (l - m + 1) \cdot 2^m. \tag{3}$$

Dla $l = 16$ i $m = 8$ oraz dla $l = 32$ i $m = 8$ liczby szablonów L_s wynoszą odpowiednio $9 \cdot 256$ i $25 \cdot 256$. Liczby te są zbyt wielkie do zilustrowania idei proponowanego algorytmu. W związku z tym w Tabeli 1 przedstawiono przykład tablicy szablonów **T** dla $l = 6$ i $m = 4$.

Tabela 1. Tablica szablonów **T** dla $l = 6$ i $m = 4$

i	***m* ustalonych bitów**	***T*[*i*,1]**	***T*[*i*,2]**	***T*[*i*,3]**
1	0000	0000**	*0000*	**0000
2	0001	0001**	*0001*	**0001
3	0010	0010**	*0010*	**0010
4	0011	0011**	*0011*	**0011
…				
14	1101	1101**	*1101*	**1101
15	1110	1110**	*1110*	**1110
16	1111	1111**	*1111*	**1111

Przyjęto, że przykładowy zbiór wzorców własnych (zaufanych fragmentów programu) **S** jest następujący:

$$\mathbf{S} = \{101110, 101101, 101100, 101011, 101010\}. \tag{4}$$

Na podstawie tablicy **T** i zbioru **S** powstaje pochodna tablica $\mathbf{T_a}$. Tablica $\mathbf{T_a}$ składa się z szablonów własnych i obcych. Jeżeli między szablonem w tablicy **T** i co najmniej jednym wzorcem własnym ze zbioru **S** co najmniej $m = 4$ kolejne bity są dopasowane, to szablon ten uznawany jest za szablon własny i jest oznaczany jako 0 w tablicy $\mathbf{T_a}$. W przeciwnym wypadku szablon jest uznawany jako obcy i oznaczany jako 1 w tablicy $\mathbf{T_a}$.

Przykładowo, szablon **T**[14,3] = **1101 w tablicy **T** jest dopasowany do szablonu **S**(2) = 101101, więc $\mathbf{T_a}[14,3] = 0$, podczas gdy szablon **T**[2,3] = **0001 w tablicy **T** nie dopasowuje żadnego wzorca własnego ze zbioru **S**, więc $\mathbf{T_a}[2,3] = 1$.

Tabela 2. Tablica $\mathbf{T_a}$ dla $l = 6$ i $m = 4$

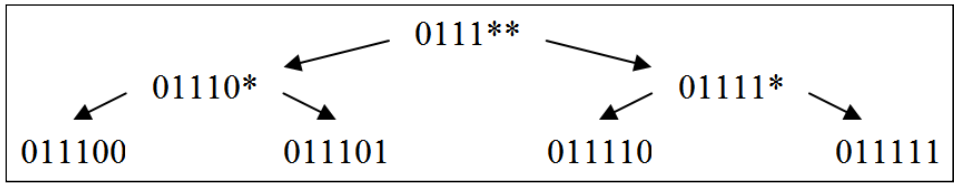
i	***m* ustalonych bitów**	$T_a[i,1]$	$T_a[i,2]$	$T_a[i,3]$
1	0000	1	1	1
2	0001	1	1	1
3	0010	1	1	1
4	0011	1	1	1
5	0100	1	1	1
6	0101	1	0	1
7	0110	1	0	1
8	0111	1	0	1
9	1000	1	1	1
10	1001	1	1	1
11	1010	0	1	0
12	1011	0	1	0
13	1100	1	1	0
14	1101	1	1	0
15	1110	1	1	0
16	1111	1	1	1

Każdy szablon obcy reprezentuje w tym przypadku cztery możliwe receptory. Przykład generacji receptora z szablonu obcego **T**[8,1] = 0111** przedstawiono na Rys. 5. Przykład ten oparty jest na drzewie binarnym. Pierwsze dwa liście zawierają 4 bity ustalone szablonu, a ich pierwsze bity nieistotne zastąpione są zerem i jedynką. W przykładzie otrzymano na tej podstawie liście 01110* i 01111*.

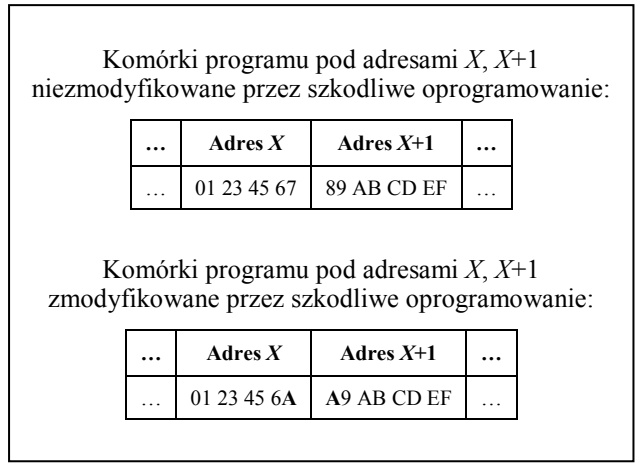
Ostatecznie, gdy pozostałe bity nieistotne „*” zastąpiono zerami i jedynkami, otrzymano cztery liście: 011100, 011101, 011110 i 011111. Liście te stają się receptorami, ponieważ na pewno nie dopasowują struktur własnych.

Na podstawie drzewa binarnego, cztery liście (receptory) szablonu obcego **T**[11,2] = *1010* są następujące: 010100, 010101, 110100 i 110101. Dwa liście (receptory) 010100, 010101 są takie same jak liście szablonu obcego **T**[6,1] i są zbędne. Całkowita liczba możliwych receptorów jest równa czterokrotności liczby szablonów obcych w $\mathbf{T_a}$. W zbiorze receptorów otrzymanych na podstawie tablicy $\mathbf{T_a}$ występuje dużo receptorów identycznych, a więc zbędnych. Potrzebna jest zatem metoda eliminacji receptorów zbędnych. W tym celu drzewo binarne każdego szablonu obcego z kolumny $\mathbf{T_a}[i,1]$ jest porównywane z drzewami binarnymi wszystkich szablonów obcych z kolumny $\mathbf{T_a}[i,2]$ i ostatecznie z drzewami binarnymi wszystkich szablonów obcych z kolumny $\mathbf{T_a}[i,3]$, eliminując zbędne gałęzie i liście. Po redukcji zbędnych gałęzi i liści zbiór liści staje się ostatecznym zbiorem receptorów **R**.

Dla zbioru wzorców własnych **S** określonego w (4), zbiór receptorów **R** = {000000, 110001, 000010, 110011, 000100, 110101, 000110, 110111, 001000, 111001, 010000, 010001, 010010, 010011, 010100, 010101, 010110, 010111, 011000, 011001, 111111}. Wszystkie 21 receptorów ze zbioru **R** wykorzystane zostanie do wykrywania anomalii.



Rys. 5. Generacja receptora na podstawie szablonu **T**[8,1] = 0111**



Rys. 6. Przykład jednobajtowej anomalii umieszczonej między 32-bitowymi komórkami

Wykorzystując metodę szablonów, system generuje receptory o długości $l = 32$ bitów z progiem aktywacji $m = 16$ bitów. Rozmiar odczytywanych komórek programu wynosi zatem 4 bajty. Metoda ta, wykorzystana w systemie z wymienionymi parametrami, nosi nazwę metody S-32 ze względu na liczbę bitów receptorów ze zbioru **R**. Zbiór receptorów ten nadaje się do wykrywania anomalii występujących w obrębie pojedynczej odczytanej czterobajtowej komórki, ale nie nadaje się do wykrywania anomalii które wystąpiły między komórkami (rozpoczynającymi się na przykład w najmłodszym bajcie komórki o adresie *X*, a kończącymi w najstarszym bajcie komórki o adresie *X*+1 – ilustruje to Rys. 6). Proponowany system zawiera więc także zmodyfikowaną wersję metody szablonów, która generuje dodatkowy zbiór receptorów $\mathbf{R_{MK}}$. Zbiór $\mathbf{R_{MK}}$ zawiera receptory o długości $l = 16$ bitów z progiem aktywacji $m = 8$ bitów wygenerowane na podstawie par kolejnych sąsiadujących ze sobą czterobajtowych komórek programu. Przy generacji tego zbioru pod uwagę brane są tylko najmłodszy bajt komórki i najstarszy bajt komórki po niej następującej. Modyfikacja metody tej nosi nazwę metody RMK (receptorów międzykomórkowych).

Należy zauważyć, że przykładowo liście drzew binarnych szablonów, które zostały odrzucone jako potencjalne receptory ze względu na dopasowanie struktury własnej na końcu programu, mogłyby nadal służyć jako receptory do wykrywania anomalii na początku programu. W związku z tym proponowany system zawiera również metodę sekcji (MS). Metoda sekcji dzieli sprawdzany program na cztery równe części (sekcje) i traktuje je jak osobne programy, co powoduje że receptory z sekcji pierwszej nie kolidują z kodem własnym z innych sekcji. W metodzie tej wykorzystano metodę szablonów 32-bitowych.

4.2. Wykrywanie anomalii

Proponowany system wykrywa anomalie w programach w obszarze chronionym korzystając z wygenerowanego wcześniej zbioru receptorów 32-bitowych **R** i opcjonalnie również zbioru receptorów 16-bitowych $\mathbf{R_{MK}}$. W przeciwieństwie do etapu generacji receptorów, na etapie wykrywania anomalii weryfikowany program odczytywany jest w jego aktualnym (obecnym) stanie. Pierwsza 32-bitowa komórka programu odczytywana jest przez system i porównywana następnie z każdym 32-bitowym receptorem ze zbioru **R**, korzystając z progu aktywacji $m = 16$. Za każdym razem, gdy między odczytaną komórką a receptorem dopasowane zostanie m kolejnych identycznych bitów z tym samym przesunięciem okna k, wykryta zostaje anomalia i zostaje ona zgłoszona użytkownikowi. Jeżeli żaden z receptorów nie jest aktywowany, odczytywana jest kolejna 4-bajtowa komórka pamięci i analogicznie porównywana jest ze wszystkimi receptorami. Proces powtarza się do momentu porównania wszystkich komórek programu ze wszystkimi receptorami.

5. Badania eksperymentalne

Proponowany system został zaimplementowany w języku C# w celach przeprowadzenia badań skuteczności opisanych metod.

Jako przykładowy plik do ochrony wykorzystano napisany w języku C++ program w formacie Win32 PE o nazwie „test.exe” i rozmiarze 6584 bajtów. Badanymi metodami były metoda losowa (L), metoda szablonów 32-bitowych (S-32), metoda szablonów 32-bitowych i 16-bitowych (RMK) i metoda sekcji (MS). Parametrami dla metod: losowej i S-32 były parametry $l = 32$, $m = 16$, dla metody RMK $l = 32$, $m = 16$ dla receptorów głównych w zbiorze **R** oraz $l = 16$, $m = 8$ dla receptorów pomocniczych w zbiorze $\mathbf{R_{MK}}$, i $l = 32$, $m = 8$ dla receptorów w metodzie sekcji, przy czym liczba sekcji jest równa 4. Metody sprawdzono dla losowo wprowadzonych anomalii o rozmiarach z zakresu [1 B; 8 B], przy czym dla każdego rozmiaru anomalii przeprowadzono 100 prób wykrycia. Wyniki badań przedstawiono w poniższych tabelach. Oznaczenia w tabelach są następujące: *NRA* – największa liczba aktywowanych receptorów dla 100 prób, *CWavg* – średni czas wykrycia anomalii w milisekundach, *W* – wykrywalność anomalii w procentach.

Tabela 3. Metoda losowa, R_{max} = 1000, R_n = 927

Rozmiar anomalii [B]	***NRA***	***CWavg***	***W***
1	1	1046	10,00%
2	1	1030	20,00%
3	1	1041	40,00%
4	1	1020	20,00%
5	1	1025	30,00%
6	3	1037	20,00%
7	2	1050	20,00%
8	2	1045	20,00%
Średnia	1,5	1036,75	22,50%

Tabela 4. Metoda losowa, R_{max} = 5000, R_n = 4693

Rozmiar anomalii [B]	***NRA***	***CWavg***	***W***
1	1	5396	30,00%
2	2	5321	50,00%
3	2	5313	40,00%
4	3	5311	70,00%
5	4	5298	90,00%
6	4	5264	60,00%
7	4	5348	80,00%
8	5	5211	60,00%
Średnia	3,125	5307,75	60,00%

Tabela 5. Metoda losowa, R_{max} = 10000, R_n = 9427

Rozmiar anomalii [B]	***NRA***	***CWavg***	***W***
1	2	10072	40,00%
2	3	10017	40,00%
3	2	10020	90,00%
4	3	10072	60,00%
5	4	10063	90,00%
6	3	9952	100,00%
7	3	10216	100,00%
8	5	9995	100,00%
Średnia	3,125	10050,875	77,50%

Tabela 6. Metoda S-32, R_n = 32

Rozmiar anomalii [B]	***NRA***	***CWavg***	***W***
1	4	12	30,00%
2	3	22	50,00%
3	8	8	20,00%
4	2	13	30,00%
5	14	21	50,00%
6	6	26	60,00%
7	6	26	60,00%
8	8	35	80,00%
Średnia	6,375	20,375	47,50%

Tabela 7. Metoda RMK, $R_n = 46$

Rozmiar anomalii [B]	*NRA*	*CWavg*	*W*
1	4	12	30,00%
2	3	22	50,00%
3	11	8	40,00%
4	11	13	70,00%
5	14	21	50,00%
6	6	26	60,00%
7	6	26	60,00%
8	11	8	100,00%
Średnia	8,25	17	57,50%

Tabela 8. Metoda sekcji, $R_n = 780$

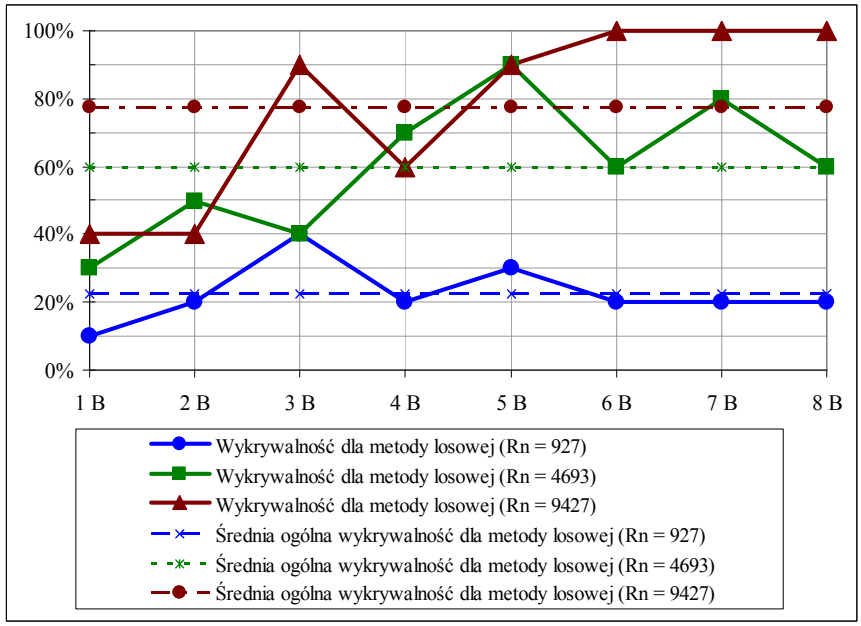
Rozmiar anomalii [B]	*NRA*	*CWavg*	*W*
1	5	10	60,00%
2	7	17	100,00%
3	9	16	90,00%
4	11	20	100,00%
5	13	20	100,00%
6	15	17	100,00%
7	24	16	100,00%
8	20	19	100,00%
Średnia	13	16,875	93,75%

Ponieważ generacja receptorów może odbyć się tylko dla niezmodyfikowanego programu, algorytm generacji musiał wykonać się tylko raz dla metod losowej, S-32 i RMK oraz 4 razy dla metody sekcji. Czas generacji receptorów dla metody losowej i $R_{max} = 1000$ wyniósł 1 s, dla $R_{max} = 5000$ wyniósł 5 s, a dla $R_{max} = 10000$ wyniósł 10 s. Czas generacji receptorów dla metody S-32 wyniósł 140 s, dla metody RMK 175 s, a dla metody sekcji 290 s.

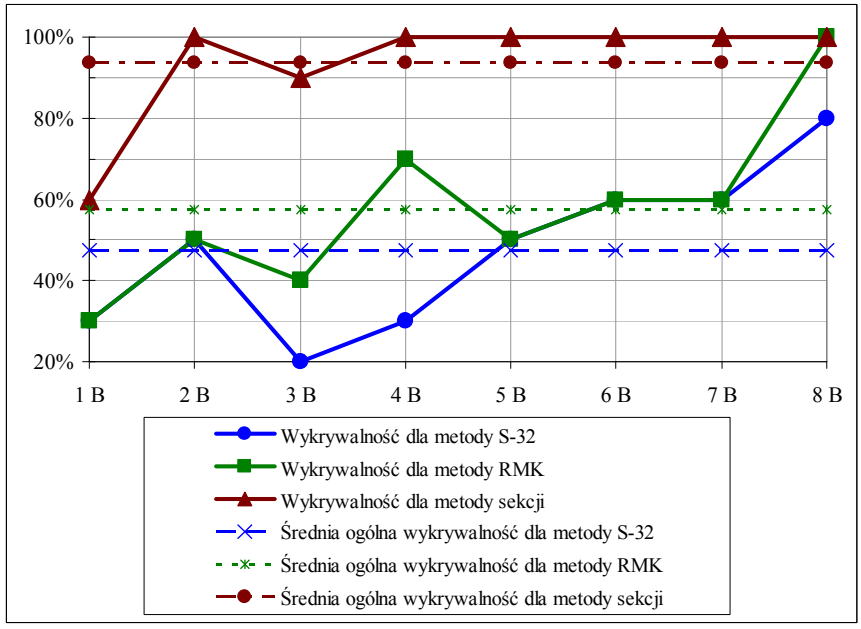
6. Analiza porównawcza

Dla sprawdzanego pliku „test.exe” system generował receptory metodą losową w tempie 1003,1 receptorów na sekundę, podczas gdy generacja metodą S-32 odbywała się w tempie 0,23 receptora na sekundę. Mimo większej liczby receptorów gotowych do wykrycia anomalii metoda losowa miała dla R_{max} = 1000 niższą wykrywalność na poziomie 22,5% w porównaniu do metody szablonów 32-bitowych, która uzyskała średnią wykrywalność na poziomie 47,5%. Zwiększenie liczby maksymalnej receptorów dla metody losowej do R_{max} = 5000 podniosło wykrywalność metody losowej do 60%, jednak spowodowało to również zwiększenie zajętości pamięci operacyjnej systemu ze względu na konieczność przechowania większej liczby receptorów. Implementacja 16-bitowych receptorów pomocniczych w proponowanej metodzie RMK podniosła średnią wykrywalność anomalii w programie o 10 punktów procentowych (uzyskując wynik 57,5%) i obniżyła średni czas wykrycia anomalii z 20,4 do 17 ms. Zajętość pamięci w tym przypadku zwiększyła się o dodatkowe 14 receptorów 16-bitowych w zbiorze $\mathbf{R_{MK}}$. Można zauważyć różnicę w zajętości pamięci między metodą losową a metodą RMK. Metoda losowa dla R_{max} = 5000 wykorzystała 4693 czterobajtowe komórki pamięci operacyjnej, a metoda RMK wykorzystała jedynie 46 takich komórek. Średni czas wykrycia anomalii dla metody losowej wyniósł 5307,75 ms, a dla metody RMK wyniósł 17 ms. Czasy generacji receptorów dla metody losowej i metody RMK wyniosły odpowiednio 5 s i 120 s. Średnia wykrywalność tych metod dla zadanych parametrów była porównywalna. Można zauważyć, że metoda losowa bardzo szybko generuje potrzebne receptory, jednak jest ich więcej i zajmują one więcej pamięci niż w innych metodach. Powoduje to, że średni czas wykrycia anomalii jest większy niż w innych metodach, gdyż receptorów do sprawdzenia jest więcej. Zwiększając liczbę maksymalną receptorów do R_{max} = 10000 uzyskano średnią wykrywalność na poziomie 77,5%. Dzieląc plik na cztery fragmenty, uzyskano w metodzie sekcji wykrywalność na poziomie 93,75%. Liczba receptorów wzrosła wtedy jednak z 32 do 780, co zwiększyło zajętość pamięci i wydłużyło czas generacji receptorów o 150 s. Można zauważyć, że dla R_{max} = 1000 w metodzie losowej rozmiar anomalii przestaje mieć wpływ na wykrywalność już powyżej 3 B, gdyż oscyluje ona w okolicach 30% mimo coraz większej liczby bajtów anomalii. W porównaniu, wykrywalność metody S-32 rosła w miarę wzrostu rozmiaru anomalii – dla 4 B wykrywalność wyniosła 30%, a dla 8 B już 60%. Rozmiar anomalii ma więc mniejsze znaczenie w metodzie losowej niż w metodach wykorzystujących szablony.

Wykresy wykrywalności metod w funkcji rozmiaru anomalii przedstawiono na Rys. 7 i Rys. 8.



Rys. 7. Wykrywalność dla metody losowej w funkcji rozmiaru anomalii



Rys. 8. Wykrywalność metod opartych na szablonach w funkcji rozmiaru anomalii

7. Wnioski

Proponowany system wykorzystuje algorytmy sztucznych systemów immunologicznych w celu wykrycia anomalii w programie. Algorytm negatywnej selekcji zaimplementowano w dwóch różnych metodach: losowej i szablonów. Implementację algorytmów przetestowano poprzez serię badań eksperymentalnych, a następnie badania te porównano. Badania wskazują, że metoda losowa jest bardzo szybka w generowaniu receptorów (trwało to maksymalnie 5 s), w porównaniu z metodą szablonów w której czas generacji receptorów wyniósł do 290 s. Wyższy czas generacji receptorów w metodach szablonowych spowodował jednak wzrost jakości samych receptorów. Duża liczba receptorów w przypadku metody losowej powodowała również wysoką w porównaniu z metodami szablonowymi zajętość pamięci operacyjnej i dłuższy czas wykrycia anomalii.

W przypadku gdy minimalizacja zajętości pamięci nie ma priorytetu nad potrzebą szybkiego rozpoczęcia pracy systemu, metoda losowa może okazać się odpowiednia. Jeżeli istotny jest natomiast czas wykrywania anomalii, odpowiedniejsza w użyciu może być metoda szablonów. Najskuteczniejsza pod względem wykrywalności dla zadanych parametrów okazała się metoda sekcji, która osiągnęła poziom 93,75%. Dalsze prace nad badaniami mogą uwzględniać na przykład zachowanie współczynnika wykrywalności w zależności od liczby bitów receptora i progu aktywacji.

Literatura

1. Somayaji A., Forrest S., Hofmeyr S., Longstaff T., *A sense of self for unix processes*, w: *IEEE Symposium on Security and Privacy*, 1996, s. 120-128.
2. Somayaji A., Hofmeyr S., Forrest S., *Principles of a computer immune system*, w: *New Security Workshop*, Langdale, Cumbria 1997, s. 75-82.
3. Forrest S., Perelson A.S., Allen L., Cherukuri R., *Self-nonself discrimination in a computer*, w: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, 1994, No 202.
4. Kephart J., *A biologically inspired immune system for computers*, w: *Fourth International Workshop on Synthesis and Simulation of Living Systems, Artificial Life IV*, 1994, s. 130-139.
5. Dasgupta D., *Immunity-based intrusion detection systems: a general framework*, w: *22nd National Information Systems Security Conference (NISSC)*, 1999.
6. Andrews P.S., Timmis J., *Tunable detectors for artificial immune systems: from model to algorithm*, w: *Bioinformatics for Immunomics*, Springer, New York, NY, USA 2010, vol. 3, s. 103-127.
7. Sobh T.S., Mostafa W.M., *A cooperative immunological approach for detecting network anomaly*, w: *J Applied Soft Computing*, 2011, vol. 11(1), s. 1275-1283.

8. Wang D., Zhang F., Xi L., *Evolving boundary detector for anomaly detection,* w: Expert *Systems with Applications*, 2011, vol. 38(3), s. 2412-2420.
9. Powers S.T., He J., *A hybrid artificial immune system and self organizing map for network intrusion detection,* w: *Information Sciences*, 2008, vol. 78(15), s. 3024-3042.
10. Li G.Y., Guo T., *Receptor editing-inspired real negative selection algorithm*, w: *Computer Science*, 2012, vol. 39, s. 246–251.
11. Laurentys C.A., Ronacher G., Palhares R.M., Caminhas W.M., *Design of an artificial immune system for fault detection: a negative selection approach*, w: Expert *Systems with Applications*, 2010, vol. 37(7), s. 5507–5513.
12. Fanelli R., *A hybrid model for immune inspired network intrusion detection*, Springer-Verlag, Phuket, Thailand 2008.
13. Mostardinha P., Faria B.F., Zúquete A., Vistulo de Abreu F., *A negative selection approach to intrusion detection*, w: Coello, C.A.C., Greensmith, J., Krasnogor, N., Liò, P., Nicosia, G., Pavone, M., *Artificial Immune Systems, Lecture Notes in Computer Science*, 2012, vol. 7597, s. 178-190.
14. Wierzchoń S.T., *Generating optimal repertoire of antibody strings in an artificial immune system*, Intelligent Information Systems, 2000, s. 119-133.

Streszczenie

Ochrona systemu operacyjnego przed infekcjami wirusowymi jest zagadnieniem, nad którym od kilku dekad pracują projektanci oprogramowania antywirusowego. Rosnąca w ostatnich latach złożoność szkodliwego oprogramowania skłoniła naukowców do poszukiwania inspiracji w rozwiązaniach naturalnych, takich jak układ immunologiczny ssaków. W artykule przedstawiono system wykrywania intruzów w systemie operacyjnym wykorzystujący algorytm negatywnej selekcji. Algorytm ten wykorzystuje ciągi binarne zwane receptorami do wykrywania zmian w chronionych programach. W systemie zaimplementowano dwie metody generacji receptorów: metodę losową i metodę szablonów. Metody te zostały przetestowane eksperymentalnie. Wyniki działania metod przeanalizowano i porównano, a następnie wyciągnięto wnioski.

Abstract

Protection of the operating system against virus infections is an area of research which has been worked on by antivirus software designers since several decades. Increasing malware complexity led scientists to seek inspiration in natural solutions, such as the mammal immune system. In the article, an intrusion detection system has been proposed. The system's inner workings are based on the negative selection

algorithm. The algorithm uses binary strings called receptors to detect modifications in the protected programs. In the system, two receptor generation methods have been presented: the random generation method and the template generation method. The methods have been tested experimentally. The results of both methods have been analysed and compared, and conclusions have been drawn.

Keywords: intrusion detection systems, artificial immune systems, viruses, malware, negative selection algorithm, receptor generation, anomaly, anomaly detection.