On the Appropriateness of Negative Selection defined over Hamming Shape-Space as a Network Intrusion Detection System

Thomas Stibor
Darmstadt University of Technology
Hochschulstr. 10
64289 Darmstadt, Germany
stibor@sec.informatik.tu-darmstadt.de

Jonathan Timmis
Departments of Electronics and Computer Science
University of York, Heslington, York
jt517@ohm.york.ac.uk

Claudia Eckert
Darmstadt University of Technology
Hochschulstr. 10
64289 Darmstadt, Germany
eckert@sec.informatik.tu-darmstadt.de

Abstract- Artificial immune systems have become popular in recent years as a new approach for intrusion detection systems. Indeed, the (natural) immune system applies very effective mechanisms to protect the body against foreign intruders. We present empirical and theoretical arguments, that the artificial immune system negative selection principle, which is primarily used for network intrusion detection systems, has been copied to naively and is not appropriate and not applicable for network intrusion detection systems.

1 Introduction

The immune system is a fascinating complex system which interacts with different organs and other complex systems e.g. the brain, to protect the body against diseases and infections. To accomplish this task, many different dynamic and adapting techniques — e.g. pattern classification, detection of new unseen intruders, memory for a second fast immune respond — are performed [1]. In recent years, immune system mechanisms have been abstracted by computer scientists for solving computational and information processing problems and a new field termed artificial immune system has arisen. In this paper, we theoretically investigate the applicability and appropriateness of the artificial immune system negative selection principle as an intrusion detection system. Previous investigations [2] based on simulations revealed that this approach is inapplicable for intrusion detection. Our investigations confirm and support these results. To present our arguments, the work is organized as follows : In section 2, anomaly and ruled based misuse models are described and examples are provided. In section 3, the immune system is briefly described, with focus on the *negative selection* principle. In section 4, the negative selection algorithm and the r-chunk matching rule are described. Additionally, the "known negative selection" problems [3] are highlighted and problems with non-detectable elements (termed holes) are illustrated. In section 5, formulas to calculate the number of generable r-chunk detectors and the number of holes are presented. These formulas are then empirically verified. Additionally, the complexity of the most known algorithms to generate r-contiguous and r-chunk detectors are presented. In section 6, the shown arguments, formulas and problems of the negative selection principle are discussed with regard to a network intrusion detection system.

2 Intrusion Detection Systems

Intrusion detection systems (short IDSs) [4, 5] are software and hardware systems that automate the process of monitoring the events occurring in a computer system or network and analyze them for signs of *intrusions*. Heady et al. [6] defined an *intrusion* as "any set of actions that attempt to compromise the integrity, confidentially and availability of information resources". Intrusions are caused by attackers accessing the system, authorized users of the systems who attempt to gain additional privileges for which they are not authorized and computer worms and viruses which carry malicious code. IDSs are based on the belief that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable. Typically, IDSs employ anomaly and ruled based misuse models in order to detect intrusions and are differenced in host-based and network-based systems. Host-based systems employ the host operating system's audit trails as the main source of input to detect intrusive activity, while network-based IDSs build their detection mechanism on monitored network traffic. One of the most popular network-based IDS is the open source program *Snort* [7]. Snort is a rule-based network IDS which contains a database, where known malicious patterns (termed signatures) are stored. Each network packet, which is monitored by Snort, is disassembled in several distinct packet components and compared to the signatures in the database. When a signature matched with a packet component, an event is triggered and appropriate actions can be executed. The following examples show different Snort rules, to detect a DDoS[1] communication, a worm and a buffer overflow attack.

[1]Distributed Denial of Service

Example 1
```
alert icmp
  $EXTERNAL_NET any -> $HOME_NET any
 (msg:"DDOS Stacheldraht client spoofworks";
 icmp_id:1000; itype:0; content:"spoofworks";
 classtype:attempted-dos; sid:227; rev:6;)
```

Example 2
```
alert udp $EXTERNAL_NET any ->
$HOME_NET 1434 (msg:"MS-SQL Worm propagation
attempt"; content:
"|81 F1 03 01 04 9B 81 F1 01|";
classtype:misc-attack; sid:2003; rev:8;)
```

Example 3
```
alert tcp $EXTERNAL_NET any ->
  $HOME_NET 143
  (msg:"IMAP partial body buffer overflow
  attempt";content:"PARTIAL"; nocase;
  content:"BODY["; distance:0; nocase;
  pcre:"/\sPARTIAL.*BODY\[[^\]]{1024}/smi";
  classtype:misc-attack; sid:1755; rev:14;)
```

In example 1, a snort signature is shown, which indicating the presence of a variant of the Stacheldraht DDoS tool [8]. Stacheldraht is a distributed denial of service tool, uses a tiered structure of compromised hosts to coordinate and participate in a denial of service attack. There are "handler" hosts that are used to coordinate the attacks and "agent" hosts that launch the attack. Communication between the handler and the agent is conducted using icmp_echoreply. The communication information is located inside an ICMP packet and consists of the ASCII string "spoofworks". Example 2 shows the snort signature in hexadecimal representation of the "well known" SQL worm, which infected million of computers, where an un-patched Microsoft SQL database was running. Example 3 shows the snort signature of a buffer overflow exploit to an IMAP Server. This event is generated when a remote authenticated user sends a malformed request for partial mailbox attributes to an IMAP server. Examples 1,2,3 emphasize, the fact that it is necessary to inspect the network packet payload, to recognize and determine the *type* of the intrusion.
In contrast, IDSs which employ anomaly models, establish profiles of normal activities of the operating system or the network traffic and detect intrusions by identifying significant deviations from the observed profiles. Network-based IDSs establish profiles based on connection vectors. A connection vector consists of different fields which characterize a network packet and the established connection such as source, destination, length of the message, time it was sent, the frequency of the communication, etc. In [5] a connection vector is shown and detailed described, which encompasses 15 fields with characteristics about the network packet and the connection, but without payload information. Ideally a combination of anomaly and ruled based misuse model is applied, because both models have drawbacks. A ruled based misuse model cannot detect attacks for which it has no signatures — they do not react well to the unknown. Anomaly based models have the weakness of high false alarm rates, i.e. "normal" is recognized as an intrusion.

3 Immune System

The immune system [9] is responsible to protect the body against disease and infections caused by pathogens. Pathogens are foreign substances like bacteria, fungi, parasites and viruses, which continuously attack the body and can lead to death in the worst case. To recognize and destroy these substances, the immune system maintains different types of cells, which cooperate in a recognition and destruction process. One type of these cells are lymphocytes, which belong to the class of white blood cells. Lymphocytes carry antibodies on their surface, which are comparable to detectors and are able to recognize pathogenic patterns (termed antigens). Lymphocytes are differentiated in two different classes. B-Lymphocytes which mature in the bone marrow and T-Lymphocytes which mature in the thymus. Both lymphocyte types are able to recognize a wide spectrum on antigens, this includes also cells which belongs to the body (termed self). To avoid this kind of self-recognition, the *negative selection* process eliminates self-reactive lymphocytes by a controlled cell death (apoptosis). The lymphocytes which survive this process are self-tolerant and recognize antigens which not belongs to the body (termed non-self).

3.1 Artificial Immune System

An artificial immune system (AIS) is a paradigm inspired by the immune system and is used for solving computational and information processing problems. AISs exploit principles and methods developed by (theoretical) immunologist and implement these in computational systems [1]. An AIS can be described and developed using a framework (see Fig. 1) which contains the following basic elements:

- A representation for the artificial immune elements;
- A set of functions, which quantifies the interactions of the artificial immune elements;
- A set of algorithms which based on observed immune principles and methods.

In the last 10 years many AIS algorithms and applications based on the immune system metaphors have been proposed. One such AIS algorithm is *negative selection*, which is mainly used for anomaly detection [10, 11] and intrusion detection systems [12, 13].

4 Negative Selection Principle

The negative selection principle is a mechanism of the immune system to protect the body against self

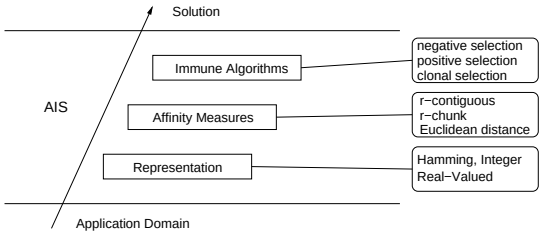
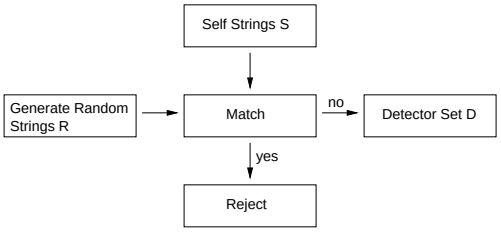
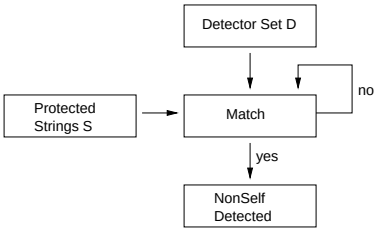


Figure 1: The AIS framework proposed by de Castro and Timmis



(a) Generation of detector set



(b) Monitor protected strings for manipulation

Figure 2: Negative selection algorithm proposed by Forrest et al.

reactive lymphocytes. This principle inspired Forrest et al. [11] to propose a negative selection algorithm to detect data manipulation caused by computer viruses. The basic idea is to generate a number of detectors in the complementary space and then to apply these detectors to classify new (unseen) data as self (no data manipulation) or non-self (data manipulation). The negative selection algorithm proposed by Forrest et al. is illustrated in figure 2 and summarized in the following steps.

Algorithm 1
Given a shape-space U, self set S and non-self set N, where

$$U = S \cup N \quad and \quad S \cap N = \emptyset.$$

1. Define self as a set S of elements of length l in shape-space U.
2. Generate a set D of detectors, such that each fails to match any element in S.
3. Monitor S for changes by continually matching the detectors in D against S.

This original algorithm has some drawbacks. First, the algorithm is inefficient, since a vast number of *randomly* generated detectors need to be discarded, before the required number of suitable once are obtained — this is a simple random search. And second, the algorithm is defined over a shape-space which induces additional problems, discussed in the following section.

4.1 Shape-Space and Affinity

The notion of *shape-space* was introduced by Perelson and Oster [14] and allows a quantitative affinity description between antibodies and antigens. More precisely, a shape-space is a metric space with an associated distance (affinity) function. A detailed overview of other shape-shapes and affinity functions used in artificial immune systems is provided in [1].

4.2 Hamming Shape-Space and R-chunk Matching

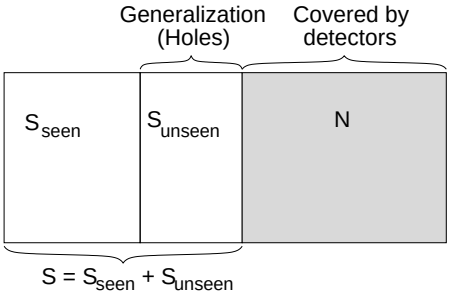
The Hamming shape-space U_l^Σ is built out of all elements of length l over an finite alphabet Σ. In the original negative selection algorithm proposed by Forrest et al. [11] it is defined over the binary alphabet $\Sigma = \{0, 1\}$. The *r-contiguous* [15] matching rule was applied to determine the affinity between a detector and an element. Informally, two elements, with the same length, match if at least r contiguous characters are identical. In succeeding works [16, 17] the performance of different matching rules over the binary alphabet are compared and the *r-chunk* [17] matching rule achieved the highest matching performance compared to the other matching rules over the binary alphabet. The r-chunk matching rule is an improved variant of the r-contiguous matching rule and is defined as follows :

Given a shape-space U_l^Σ, which contains all elements of length l over an alphabet Σ and a shape-space D_r^Σ.

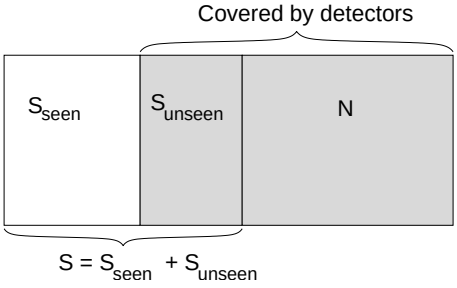
Definition 1 *An element $e \in U_l^\Sigma$ with $e = e_1 e_2 \dots e_l$ and detector $d \in \mathbb{N} \times D_r^\Sigma$ with $d = (p, d_1 d_2 \dots d_r)$, for $r \leq l$, $p \leq l - r + 1$* match with r-chunk rule *if $e_i = d_i$ for $i = p, \dots, p + r - 1$.*
Informally, element e and detector d match if a position p exists, where all characters of e and d are identical over a sequence length r.
The Hamming shape-space and the affinity functions r-contiguous and r-chunk seem very appealing at first sight, because it allows a straightforward mathematical analysis [18] and properly abstracts the binding between antibody and antigen [15].

4.3 Generalization by Undetectable Elements

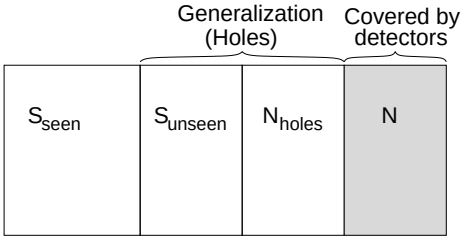
All matching rules, including the r-chunk rule investigated in [17], cause undetectable elements (termed holes). Holes are elements of N or self elements not seen during the training phase. For these elements no detectors can be generated and therefore, they cannot be recognized and classified as non-self elements. The



(a) The detector set generalizes well, as all unseen self elements S_{unseen} (holes) are classified as self and the rest as non-self .



(b) The detector set overfits, because no holes exist and therefore unseen self elements S_{unseen} are classified as non-self e elements.



(c) The detector set underfits, because a larger number of non-self elements N_{holes} are not recognized by the detectors and therefore are classified as self.

Figure 3: Holes are necessary to generalize beyond the training set. Too many holes results in an underfitting, whereas, no holes results in an overfitting.

term holes is an improper expression, because holes are *necessary*, to generalize beyond the training set. A detector set which generalizes well, ensures that seen and unseen self elements are *not* recognized by any detector, whereas all other elements are recognized by detectors and classified as non-self (see Fig. 3(a)). A detector set which covers all non-self elements and all unseen self elements *overfits*, because no holes (no generalization) exists (see Fig. 3(b)). In contrast, a large number of holes implies, that a large number of unseen self elements and non-self elements as well, are not covered by the detector set and therefore the detector set *underfits* (see Fig. 3(c)). Balthrop et al. [16] proposed a method (termed crossover-closure) to find holes for the r-chunk matching rule by given parameters l, r and S. We have summarized Balthrop's method algorithmically (see algorithm 2) and show (see Fig. 4) an illustrative example of the construction

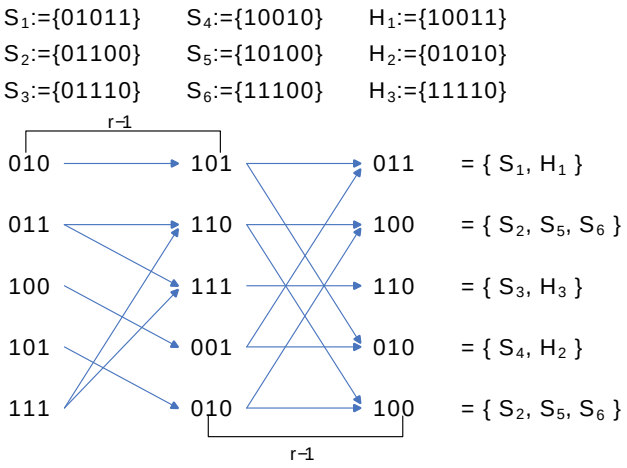


Figure 4: Construction to find holes for $r = 3$ and $l = 5$ for the r-chunk matching rule for $S = \{01011, 01100, 01110, 10010, 10100, 11100\}$. For elements $H_1 = \{10011\}, H_2 = \{01010\}, H_3 = \{11110\}$ it is not possible to generate a detector which recognize H_1, H_2, H_3. All possible generable detectors are $D = \{\{1|000\}, \{1|001\}, \{1|110\}, \{2|000\}, \{2|011\}, \{2|100\}, \{3|000\}, \{3|001\}, \{3|101\}, \{3|111\}\}$

for a given set S and $l = 5, r = 3$.

Algorithm 2
Given a set $S = \{S_1, S_2, \ldots, S_n\}$ of self strings, element length l and matching length r :

1. Cut S_i in $l - r + 1$ substrings $S_{i,j} := S_i[j, \ldots, r - 1 + j]$
   for $j = 1, \ldots, l - r + 1$, $i = 1, \ldots, n$.
2. Connect substring $S_{i,j}$ to $S_{k,j+1}$ with a direct edge, if the last $r - 1$ characters of $S_{i,j}$ and the first $r - 1$ characters of $S_{k,j+1}$ are identical, for $i = 1, \ldots, n$, $k = 1, \ldots, n$ and $j = 1, \ldots, n - 1$.
3. Traverse and shuffle coincident substrings $S_{i,1}, \ldots, S_{i,l-r+1}$ for $i = 1, \ldots, n$ to obtain the set S of self strings and the set H of undetectable elements[2].

Algorithm 2 shows, that holes arise in commonly occurring distinct self strings. In the next section, we show formulas to calculate the number of generable detectors and the number of holes under the assumption that S is randomly drawn from U_l^{Σ}.

5 Number of Generable Detectors and Resulting Holes

The number of generable r-chunk detectors were investigated in [19, 20]. In these works, formulas are provided to calculate the number of generable detectors by given the parameter $|S|, l, r$. Esponda et al. [19] estimated the total number of all generable detectors by

[2]To obtain set H, perform the set difference $(H \cup S) \backslash S$

given $|S|, l, r$ as follows :

$$|D|=(l-r+1)(2^r-E(r,|S|))=\left(1-\tfrac{1}{2^r}\right)^{|S|}\cdot(l-r+1)\cdot 2^r \quad (1)$$

The term $E(r,|S|) \approx 2^r - 2^r(1-2^{-r})^{|S|}$ approximates the expected number of distinct patterns by given the r-chunk length r and the cardinality S.

Stibor et al. [20] have shown and proved a more general formula to calculate the numbers of all generable detectors :

$$|D|=\left(1-\tfrac{1}{(l-r+1)\cdot|\Sigma|^r}\right)^{|S|\cdot(l-r+1)}\cdot(l-r+1)\cdot|\Sigma|^r \quad (2)$$

The terms (1) and (2) only distinguish in the expression $(l-r+1)$ and the generalization of the alphabet size Σ. A simple calculation shows that the difference between term (1) and (2) is negligible.

In the work [19], Esponda et al. additionally provided formulas to calculate the number of holes for the binary alphabet $\Sigma = \{0,1\}$. In order to produce these formulas, they employed the crossover closure CC construction (see Fig. 4 and algorithm 2) and approximates the number of outgoing edges, which results in the numbers of holes and self strings. Summarizing Esponda et al. [19] derivation, the number of holes $|H|$ can be calculated as follows :

$$|H|=CC(l,r)-|S| \quad (3)$$

where

$$CC(l,r)=E(r,|S|)\cdot(1+P(r,|S|))^{(l-r)} \quad (4)$$

The subterm (4) yields the number of all strings, which can be constructed by the crossover closure. The crossover closure also constructs self strings S (see Fig. 4) and therefore, this proportion must be subtracted (see Eq. (3)). The subterm $P(r,|S|)$ results from the likelihood of a substring s_i having outgoing edges.

$$P(r,|S|)=1-\tfrac{1}{2}(P_0+P_1)$$

where

$$\begin{aligned} P_0 &= \left(1-\tfrac{1}{2^{r+1}}\right)^{4|S|} \\ P_1 &= 4\left(1-\tfrac{1}{2^{r+1}}\right)^{2|S|}-4\left(1-\tfrac{1}{2^{r+1}}\right)^{3|S|} \end{aligned}$$

Where P_0 is the probability of a substring to have no outgoing edges and P_1 to have one outgoing edge (see [19] for more details). Using the subterm $P(r,|S|)$ and simplifying term (4) one obtains

$$H(|S|,l,r)=T_1\cdot T_2-|S| \quad (5)$$

where

$$\begin{aligned} T_1 &= 2^r-2^r\left(1-\tfrac{1}{2^r}\right)^{|S|} \\ T_2 &= \left[2-\tfrac{1}{2}\left(1-\tfrac{1}{2^{r+1}}\right)^{4|S|}-2\left(\left(1-\tfrac{1}{2^{r+1}}\right)^{2|S|}-\left(1-\tfrac{1}{2^{r+1}}\right)^{3|S|}\right)\right]^{(l-r)} \end{aligned}$$

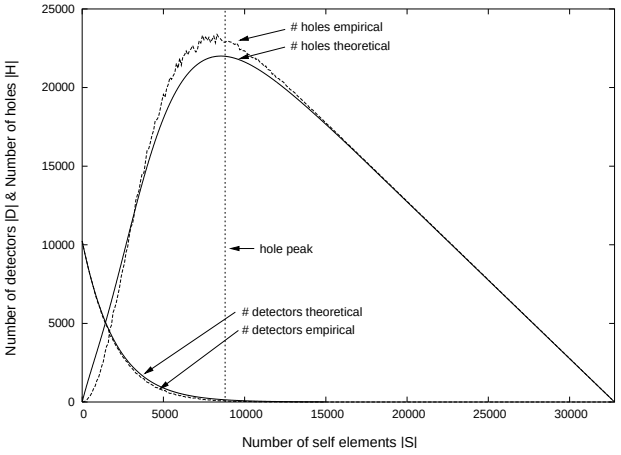


Figure 5: Numbers of detectors and numbers of holes, for $l = 15, r = 11, |\Sigma| = 2$ and $|S| = \{0,\ldots,2^{15}\}$, calculated with Eqs. (2) and (5) and empirically by the algorithm output

5.1 Empirical Formula Verifications

We have implemented the algorithm proposed by Stibor et al. [20], which generates all possible r-chunks detectors given the alphabet size Σ, string length l and r-chunk length r. With simple modifications — count the number of non-self strings not detected by any detector, the algorithm also outputs the number of holes. Since the algorithm first initializes a hashtable of size $|\Sigma|^r \cdot (l-r+1)$, we perform our simulations on a small value of l. The values $l = 15$, $r = 11$ and $|\Sigma| = 2$ were used. For each $|S| = \{0,\ldots,2^{15}\}$ (randomly determined) a detector generation (algorithm) run was performed and the resulting number of detectors and holes were depicted in a graph (see Fig. 5). To verify the accuracy of Eq. (2) which calculates the number of detectors and Eq. (5) which calculates the number of holes, these results were also depicted in the graph (see Fig. 5). One can see, that the theoretical derived formulas, approximate well the empirical results. More interesting is the fact that for $l = 15, r = 11$, the number of generable detectors exponentially decrease to the number of self elements, whereas the number of holes exponentially increase. Reaching a certain proportion of self elements, no detectors can be generated and *all* non-self elements and unseen self elements are holes. In figure (5) this is termed *hole peak*. After the hole peak is passed, the holes decrease nearly linear to 0, because the number of self elements increase and the relation $U = S \cup N$ must hold.

5.2 Control Number of Detectors and Holes with r-chunk length r

Investigating equations (2) and (5), one can see, that the exponential curves behavior can be controlled by the parameters $|S|, l, r$. Usually, the length l is an *a-priori* defined value, which is determined by the payload length or by the length of the connection vector.

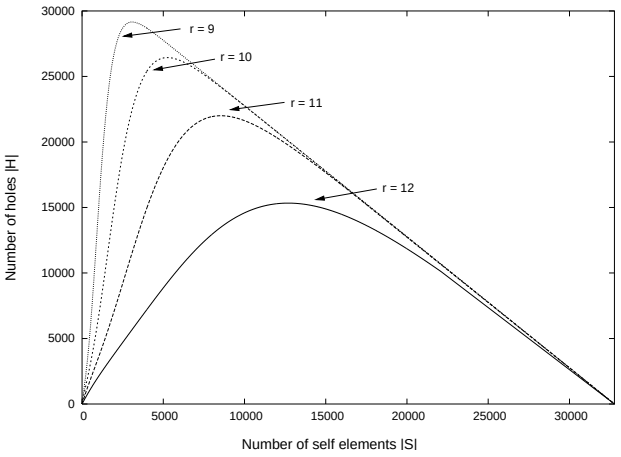


Figure 6: Number of holes for r-chunks length $r = \{9, 10, 11, 12\}$, $l = 15, |\Sigma| = 2$ and $|S| = \{0, \ldots, 2^{15}\}$

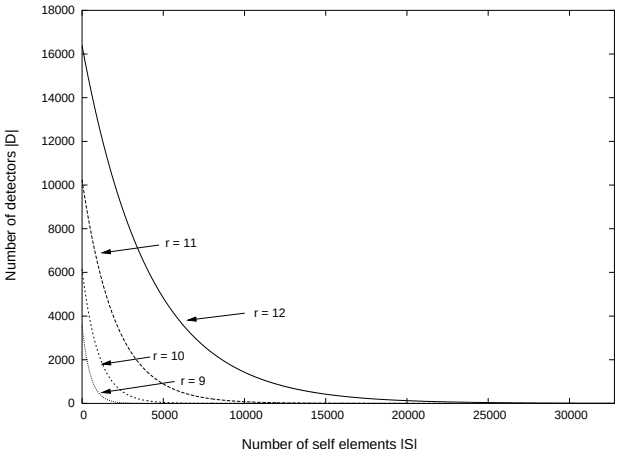


Figure 7: Number of generable detectors for r-chunks length $r = \{9, 10, 11, 12\}$, $l = 15, |\Sigma| = 2$ and $|S| = \{0, \ldots, 2^{15}\}$

The number of self elements $|S|$ depends on the element length l and is not adaptable, when the length l is *a-priori* defined. Therefore, the parameter r can be used, to control the number of generable detectors and the number of holes. In figures (6) and (7) the effects for different r-chunks lengths are depicted. One can see, that increasing r closer to l, the number of holes decrease and the number of generable detectors increase. Furthermore, the hole peak moves toward a larger amount of self elements. This is an important property, since not all self elements are seen during the training phase. When r is not close to l, the number of holes (the generalization) increase exponentially with the number of self elements until the hole peak is reached. This means that the detector set exponentially underfits.

The number of holes can also be controlled (reduced) with permutation masks [21, 19]. A permutation mask is a bijective mapping π that specifies a reordering for all strings $u \in U_l^{\Sigma}$, i.e. $u_1 \rightarrow \pi(u_1), u_2 \rightarrow \pi(u_2), \ldots, u_{|\Sigma|^l} \rightarrow \pi(u_{|\Sigma|^l})$. Permutation maps *can* reduce the number of holes, when π is appropriate chosen and a certain number of detectors are generable. In our consideration the self elements are randomly determined. Let $l = 49, r = 12, |\Sigma| = 2$ and $|S| = 2^{16}$. Using formula 2, one obtains $|D| = 0.0175$ generable detectors. A randomly chosen permutation mask π is comparable to choosing randomly $|S| = 2^{16}$ new self elements. Applying formula 2, one obtains likewise $|D| = 0.0175$ generable detectors. Applying permutation masks is *only* useful, when a certain number of detectors are generable. Figure 7 illustrates this fact. For $l = 15, |S| = 1000$ it is not possible to generate any detector for $1 \leq r \leq 10$ and therefore the permutation masks providing no benefits.

5.3 Detector Generating Algorithm with Exponential Complexity

The first version of the negative selection algorithm (see algorithm 1) [11] mirrored most closely the generation of T-Cells in the immune system. Candidate detectors were drawn at random from U and checked against all elements in S. This process of random generation and checking was repeated until the required number of detectors was generated. This random search for detectors has a constant space complexity in $|S|$, but an exponential runtime complexity in $|S|$ and is therefore not applicable. In a succeeding work, D'haeseleer et. al [22] proposed two detector generating algorithms for the r-contiguous matching rule with an improved runtime complexity. The *linear time detector generating algorithm* has a linear runtime complexity in $|S|$ and $|D|$, but still requires time and space exponential in r :

$$\begin{aligned} \text{Time}: &\quad O\left((l-r)\cdot|S|\right) + O\left((l-r)\cdot 2^r\right) + O\left(l\cdot|D|\right) \\ \text{Space}: &\quad O\left((l-r)^2\cdot 2^r\right) \end{aligned}$$

The second algorithm termed *greedy detector generating algorithm* has a similar complexity :

$$\begin{aligned} \text{Time}: &\quad O\left((l-r)\cdot|D|\cdot 2^r\right) \\ \text{Space}: &\quad O\left((l-r)^2\cdot 2^r\right) \end{aligned}$$

Another algorithm termed *binary template* for r-contiguous matching rule was proposed by Wierzchoń [23]. It has also an exponential complexity in r :

$$\begin{aligned} \text{Time}: &\quad O\left((l-r)\cdot 2^r\cdot|D|\right) + O\left(2^r\cdot|S|\right) \\ \text{Space}: &\quad O\left((l-r)\cdot 2^r\right) + O\left(|D|\right) \end{aligned}$$

For the r-chunk matching rule, Stibor et. al [20] proposed an algorithm, which is used in this work, with complexity :

$$\begin{aligned} \text{Time}: &\quad O\left((l-r)\cdot|\Sigma|^r\right) + O\left(|S|\cdot(l-r)\right) + O\left(|\Sigma|^r\right) \\ \text{Space}: &\quad O\left((l-r)\cdot|\Sigma|^r\right) \end{aligned}$$

All four algorithms have a runtime and a space complexity, which is exponential in r and are *only* applicable for small values of r. Using for example a value

$r = 64$ and an alphabet size $|\Sigma| = 2$, the space and time complexity is infeasible high. This complexity problem makes it inapplicable for network intrusion detection application.

6 AIS and Intrusion Detection Systems

In many works the appealing connections between the immune system and the IDSs are stressed — an overview is provided in [13]. Intuitively, it seems obvious to abstract immune system principles and conceptualize algorithms for intrusion detection tasks. Especially, the capability of the immune system for dynamically adapting to previously unseen disease, is an attractive property for developing intrusion detection systems, which behave in a similar manner.

We showed in section 2, how the IDS Snort recognizes attackers and worms by means of pattern matching. Usually, the IDS signatures have a length of 10 bytes or more. The signature which matches for example the IMAP buffer overflow, has a string length of 39 bytes. Intrusion detection systems which employ anomaly models, require connection vectors of a determined length which characterize the network packets and connections. Assuming each field in the connection vector requires 2 bytes to store the data[3], then the connection vector described in [5], requires 30 bytes.

Using the analysis in section 5, which shows the coherence between the number of generable detectors and the number of resulting holes. It is shown, that r *must* be close to l to generate a certain number of detectors and to control the number of holes (under/overfitting behavior). A large number of holes implies a low detection rate, i.e. attacks are not recognized by the detectors. On the other hand, a limited number of holes results in a high false alarm rate, i.e. unseen normal data is recognized by the detectors and classified as an attack. Furthermore, the complexity of the most known algorithms to generate detectors is presented and the high runtime and space complexity is stressed.

Combining these arguments, it is clear, that the so far proposed AIS intrusion detection approach based, on the negative selection principle[4] is not appropriate and not applicable for these kind of network intrusion detection models. Using a connection vector, which consist of 240 bits or signatures which consist of 80 bits, it is not possible to generate in polynomial time a certain number of detectors and to obtain a linear control on the number of holes.

7 Conclusion

This work investigated the appropriateness of the negative selection principle as a network intrusion detection technique. With mathematical arguments and empirical verifications it is shown, that the negative selection algorithm defined over the Hamming shape-space is not an appropriate method for a network-based IDS. The Hamming shape-shape and the matching rules induce non-detectable elements (termed holes) which are necessary to generalize beyond the training set. However, the number of holes increase exponentially when the r-chunk length r is not close to the element length l and therefore an exponential number of elements cannot be detected by the generated detectors. To generate a certain number of detectors and to obtain a non-exponential number of holes, r must lie near l. However, this results in an exponential space and time complexity and makes the negative selection algorithm inapplicable. Moreover, existing network intrusion detection models[5] were briefly presented. It was shown, that the negative selection principle defined over the Hamming shape-space is not an appropriate technique for both network intrusion detection models.

[3] which is very optimistic estimation, because many fields are contiguous values and require at least 4 bytes

[4] defined over the Hamming shape-space associated with the r-chunk and r-contiguous matching rule

[5] anomaly and ruled based misuse

References

[1] de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer-Verlag (2002)

[2] Kim, J. and Bentley, P. J.: Evaluating negative selection in an artificial immune system for network intrusion detection. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, San Francisco, California, USA, Morgan Kaufmann (2001) 1330–1337

[3] Freitas, A., Timmis, J.: Revisiting the Foundations of Artificial Immune Systems: A Problem Oriented Perspective. In Timmis, J., Bentley, P., Hart, E., eds.: Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS). Volume 2787 of Lecture Notes in Computer Science., Springer (2003) 229–241

[4] Bace, R., Mell, P.: Intrusion Detection Systems. National Institute of Standards and Technology (NIST). (2001) http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf.

[5] Mukherjee, B., Heberlein, L.T., Levitt, K.N.: Intrusion detection system. IEEE Network (1994)

[6] Heady, R., Luger, G., Maccabe, A., Servilla, M.: The architecture of a network level intrusion system. Technical report, Computer Science Department, University of New Mexico (1990)

[7] Koziol, J.: Intrusion Detection with Snort. Sams (2003)

[8] Dittrich, D.: The stacheldraht distributed denial of service attack tool (1999) http://staff.washington.edu/dittrich.

[9] Janeway, C.A., Travers, P., Walport, M., Shlomchik, M.: Immunologie. Spektrum Akademischer Verlag (2002)

[10] Dasgupta, D., Forrest, S.: Novelty detection in time series data using ideas from immunology. In: Proceedings of the 5th International Conference on Intelligent Systems, IEEE Computer Society Press (1996)

[11] Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press (1994)

[12] Hofmeyr, S.A., Forrest, S., D'haeseleer, P.: An immunological approach to distributed network intrusion detection. In: First International Workshop on the Recent Advances in Intrusion Detection. (1998)

[13] Aickelin, U., Greensmith, J., Twycross, J.: Immune system approaches to intrusion detection – a review. In Giuseppe Nicosia, Vincenzo Cutello, P.J.B., ed.: Proceedings of the 3nd International Conference on Artificial Immune Systems (ICARIS). Volume 3239 of Lecture Notes in Computer Science., Springer-Verlag (2004) 316–329

[14] Perelson, A.S., Oster, G.: Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-nonself discrimination. In: J. Theor. Biol. Volume 81. (1979) 645–670

[15] Percus, J.K., Percus, O.E., Perelson, A.S.: Predicting the size of the t-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination. Proceedings of National Academy of Sciences USA **90** (1993) 1691–1695

[16] Balthrop, J., Esponda, F., Forrest, S., Glickman, M.: Coverage and generalization in an artificial immune system. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, Morgan Kaufmann Publishers (2002) 3–10

[17] González, F., Dasgupta, D., Gomez, G.: The effect of binary matching rules in negative selection. In: Genetic and Evolutionary Computation – GECCO-2003. Volume 2723 of Lecture Notes in Computer Science., Chicago, Springer-Verlag (2003) 195–206

[18] D'haeseleer, P.: An immunological approach to change detection: Theoretical results. In: Proc. 9th IEEE Computer Security Foundations Workshop. (1996) 18–26

[19] Esponda, F., Forrest, S., Helman, P.: A formal framework for positive and negative detection schemes. IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics **34** (2004) 357–373

[20] Stibor, T., Bayarou, K.M., Eckert, C.: An investigation of R-chunk detector generation on higher alphabets. In: Genetic and Evolutionary Computation – GECCO-2004, Part I. Volume 3102 of Lecture Notes in Computer Science., Seattle, WA, USA, Springer-Verlag (2004) 299–307

[21] Hofmeyr, S.A., Forrest, S.: Architecture for an artificial immune system. Evolutionary Computation **8** (2000) 443–473

[22] D'haeseleer, P., Forrest, S., Helman, H.: An immunological approach to change detection: algorithms, analysis, and implications. In: Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society, IEEE Computer Society Press (1996) 110–119

[23] Wierzchoń, S.: Generating optimal repertoire of antibody strings in an artificial immune system. In: Intelligent Information Systems, Springer Verlag (2000) 119–133