Ransomware Detection Based on an Improved Double-Layer Negative Selection Algorithm

Tianliang Lu, Yanhui Du(✉), Jing Wu, and Yuxuan Bao

People's Public Security University of China, Beijing, China
duyanhui@ppsuc.edu.cn

Abstract. The encrypting ransomware using public key cryptography is almost impossible to decrypt, so early detection and prevention is more important. Signature matching technology has low detection rate for unknown or polymorphic ransomware, and some intelligent algorithms have been proposed for solving this problem. Inspired by the Artificial Immune System (AIS), an improved double-layer negative selection algorithm (DL-NSA) was proposed which can reduce the number of holes in NSA and increase the detection rate. To obtain the behavior characteristics (e.g., files read or write, cryptography APIs call and network connection) of ransomware, a Cuckoo sandbox was built to simulate the malicious code running environment. After dynamic analysis, the behavior characteristics of ransomware were encoded to antigens. The improved double-layer negative selection algorithm has two sets of immune detectors. The first layer detectors set was generated by the original negative selection algorithm using *r*-contiguous bits matching. The second layer detectors set was directional generated holes' detectors using *r*-chunk matching with variable matching threshold. Simulation result shows that comparing with NSA this algorithm can achieve high-rate space coverage for non-self, and can increase the detection rate of ransomware.

Keywords: Ransomware · Negative selection algorithm · API call sequence · Artificial Immune System · Cuckoo sandbox

1 Introduction

Ransomware is one of the most threatening attacks nowadays. In the late 1980s, PC Cyborg known as the first ransomware has been developed. In recent years, ransomware of mobile devices has grown sharply. Ransomware targeting companies or governments is also rising. Wannacry that hit the headline in May 2017, has affected more than 200,000 computers in 150 countries, with total damages ranging from hundreds of millions to billions of dollars [1].

The ransom is usually paid by snail mail, bank transfer or Amazon gift card. With the popular of cryptocurrencies, nowadays ransom commonly paid by bitcoins. From 2013 to mid-2017, the market for ransomware payments has a minimum worth of USD 12,768,536 [2]. In January 2018, a hospital Hancock Health paid out a $55,000 bitcoin ransom following a SamSam infection, because paying up was deemed the quickest

H. Gao et al. (Eds.): TridentCom 2019, LNICST 309, pp. 46–61, 2020.
https://doi.org/10.1007/978-3-030-43215-7_4

way to get systems back online. But sometimes the victims will not get the decryption key even paying the ransom.

The crypto ransomware can be most destructive typically using strong encryption algorithms [3]. Ransomware such as Wannacry utilized combined encryption algorithms of AES and RSA to make the encryption harder to decrypt [4].

The number of ransomware attacks has grown partly because attackers have adopted Ransomware as a Service (RaaS) [5]. RaaS is available over the dark web, and RaaS is enabling even the most technically illiterate cybercriminal to extort payments from victims.

In order to accurately recognize unknown or polymorphic instances, inspired by the biological immune system to eliminate bacteria, a ransomware detection method based on an improved double-layer negative selection algorithm was proposed.

The rest of this paper is organized as follows. Section 2 gives an overview of the related previous work. Section 3 describes the negative selection algorithm and the problem of undetectable holes. Section 4 introduces the framework and details of the improved double-layer negative selection algorithm. Section 5 explains the ransomware analysis environment and the extraction of behavior characteristics. Section 6 evaluates our ransomware detection algorithm through experiments. Section 7 concludes the paper.

2 Related Work

2.1 Ransomware Detection

In recent years, studies have been carried out on the detection of ransomware attack. The common malware detection methods include static detection and dynamic detection. Some researchers investigate machine learning methods for detecting viruses.

Static Detection

Static detection is the analysis of a malware performed without actually executing the program. Antivirus software mainly uses the signature-based detection method. Depending on a large number of virus signatures, antivirus software can detect known viruses, but cannot deal with unknown viruses. Opcodes are widely used for static detection. Santos et al. [6] proposed a method to detect unknown malware families based on the frequency of the appearance of the opcodes sequences. Wang et al. [7] converted the opcodes sequence to an image, and the image is compared with the image generated from the known malware sample. Zhang et al. [8] proposed a classification method of ransomware families with machine learning based on n-gram of opcodes, and *TF-IDF* is calculated to select feature n-grams which exhibit better discrimination between ransomware families.

Dynamic Detection

Dynamic detection is the analysis of a malware performed while executing the program. The suspicious program is run in a controlled environment while recording the malicious operations. Information that can be obtained by dynamic analysis is API calls, system calls, instructions traces, registry changes, files changes and network connections.

Ransomware often use packing (such as ASProtect and Themida) and obfuscation techniques to avoid being detected by static analysis tools. Therefore, dynamic analysis is indispensable to understand the main features and functionalities of ransomware [9]. Xu et al. [10] introduced a framework for hardware-assisted malware detection based on monitoring and classifying memory access patterns. Scaife et al. [11] proposed an early-warning detection system for ransomware that checks for file activities and alerts the user in case of suspicious activities.

Monitoring the API function calls can be useful for detecting ransomware attacks. Many researchers agree that ransomware's typical behavior involves the encryption of files and showing a ransom message, which can be identified through the ransomware's use of API function calls [9, 12, 13].

2.2 Artificial Immune System

Bioscience has been a source of inspiration for innovative solutions to computer science and engineering problems for many years. The latest research that has attracted widespread attention in this field is the Artificial Immune Systems (AIS). AIS is inspired by biological immunology to solve complex practical problems by simulating the functions, principles and models of the immune system [14]. It's highly distributed, adaptive, and self-organizing nature, together with learning and memory features offer rich metaphors for its artificial counterpart [15].

AIS has been applied in many research areas, especially to solve many computer security problems [16, 17], such as intrusion detection [18], malware detection [19] and spam detection [20].

3 Negative Selection Algorithm and the Problem of Holes

3.1 Negative Selection Algorithm

Negative selection algorithm [21] first proposed by Forrest, is a computational imitation of self/non-self discrimination. It is modeled off the T-cell maturing process that happens in the thymus. It has been successfully applied to anomaly detection systems. The negative selection algorithm has two phases [21]: generation of detectors set (see Fig. 1) and non-self detection (see Fig. 2).

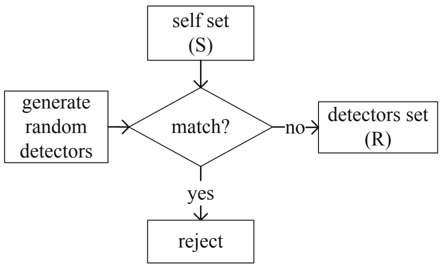

Fig. 1. Generation of detectors set

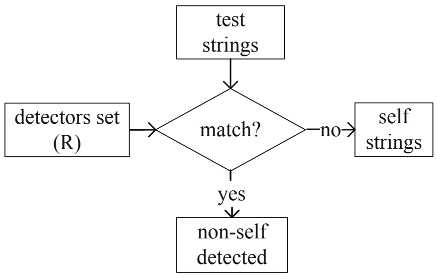

Fig. 2. Non-self detection

In NSA, elements including detectors are represented by binary strings. $U = \{0, 1\}^L$ represents all binary strings of length L. Define shape space U, self set S and non-self set N, which satisfy $U = N \cup S$ and $N \cap S = \emptyset$.

Some of the most widely used matching rules are Hamming distance, r-contiguous bits, r-chunk, etc. Take r-contiguous bits matching for example. $a = a_1 a_2 \ldots a_L$ and $b = b_1 b_2 \ldots b_L$ are two strings of length L, a matches b if and only if $\exists i \leq L - r + 1$ which meets $a_j = b_j, j = i, i + 1, \ldots, i + r - 1$.

3.2 The Problem of Undetectable Holes

There are a large number of holes exist in anomaly detection system based on negative selection algorithm, which will affect the detection rate of the algorithm. These holes are non-self individuals that cannot be recognized by all possible detectors. Holes (undetectable antigens) also exist in the biological immune system. Hofmeyr [22] proposed to use multiple representations to reduce the number of holes by using the MHC mechanism of the biological immune system. Zhang et al. [23] introduced the r-variable detection algorithm to reduce the number of holes by adjusting the matching threshold. However, this method uses a randomly generation mechanism when generating the detectors, which is inefficient for covering holes.

The main reason of holes is the partial matching rules. The matching rules adopted by negative selection algorithm need a matching threshold, which embodies the characteristics of the partial matching rules. The matched two strings do not need to be exactly the same, as long as the degree of similarity is greater than the threshold. Partial matching can be thought as an approximation or generalization.

There are two types of holes in the negative selection algorithm using r-contiguous bits matching: crossover holes and length-limit holes [24].

(1) A crossover hole is a "crossover" of certain self strings. The hole h is not in the self set S, but string h is the cross combination of certain r-step sliding windows of S. For instance (see Fig. 3), let $\{S_1, S_2\}$ = {1010, 0001} be the self set S with the string length L = 4. Define step r = 2, S_1 generates three substrings {10, 01, 10} and S_2 generates three substrings {00, 00, 01}. These six substrings can be combined together into four strings {1010, 1001, 0001, 0010} in the direction of the

arrow, in which {1001, 0010} are crossover holes, and the corresponding detectors are impossible to be generated.

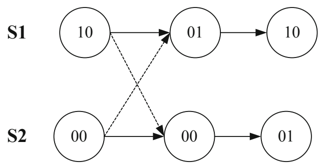

Fig. 3. Crossover holes

(2) A length-limit hole is one that has at least one window that does not exist in self strings and has some other windows that will match self strings [24]. For instance, $S = \{010, 011\}$ with string length $L = 3$ and step $r = 2$, in that case string $h = 110$ is a length-limit hole. The detectors for hole h must be generated by the template 11* or *10, where * denotes 0 or 1. But such detectors will not be generated because they match self elements.

4 An Improved Double-Layer Negative Selection Algorithm

4.1 The Double-Layer Negative Selection Algorithm (DL-NSA)

The double-layer negative selection algorithm contains two matching processes (see Fig. 4), which use the mature detector set D and the hole detector set D_H respectively.

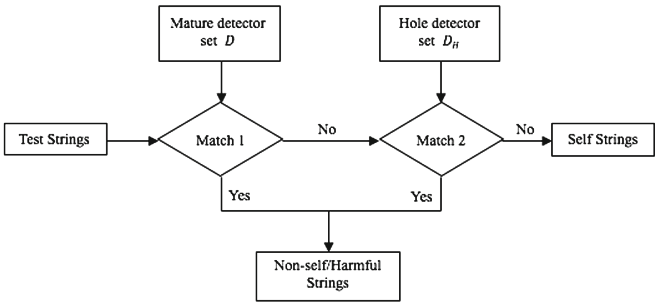

Fig. 4. The double-layer negative selection algorithm

The First Layer Matching
The set D consists of the mature detectors generated by the original negative selection algorithm, which can identify most of the non-self strings using the r-contiguous bits matching.

The Second Layer Matching

The set D_H consists of the variable-sized r-chunk detectors generated by the algorithm that will be introduced in Sect. 4.2. The hole detector set D_H is mainly used to cover holes and improve the accuracy of detection. The number of hole detectors can also be dynamically adjusted according to the hole coverage requirement.

4.2 The Hole Detectors Generation Algorithm

In the biological immune system, holes represent pathogens that cannot be recognized by the immune system, and pathogens usually evolve into holes to avoid detection by the immune system. For computer anomaly detection systems, intrusion behavior and virus programs are also evolving to seem like normal behavior and procedures. Borrowing the principle of crossover holes, the intrusion behavior makes the detection even harder by splicing a series of normal operation segments to complete the attack process. For anomaly detection systems, how to improve coverage of holes is an urgent problem to be solved.

Variable-Sized *r*-chunk Detectors

Compared with the NSA proposed by Forrest, this paper introduces variable-sized r-chunk matching detectors, which can effectively cover the holes in anomaly detection systems.

The r-chunk matching is defined as follows: a detector $d = (i, d_1 d_2 \ldots d_m)$ matches a string $a = a_1 a_2 \ldots a_n (m \le n)$ under the r-chunk matching rule, if and only if $a_j = d_j, j = i, i+1, \ldots, i+r-1 (i \le m - r + 1)$, where i represents the matching starts position.

The r-chunk matching indicates that when the detector d and the antigen a match when they have at least r identical contiguous bits from the i-th bit position. The r-chunk matching reduce the space coverage of detectors by defining start position i and matching length r, so as to solve the problem of length-limit holes. For example, the detector d = {1, 11} with r = 2 can successfully detect the length-limit hole h = 110 in Sect. 3.2.

Meanwhile, the variable-sized r-chunk detectors can cover the space that r-contiguous bits detectors cannot cover. By increasing the matching length r, detection range of the detector is further reduced to cover holes that are much closer to self elements. For example, when r = 3 the detector d = {1,100} and d = {1,001} can successfully detect the crossover holes {1001, 0010} in Sect. 3.2 respectively.

Generation Algorithm of Hole Detectors

Some scholars have proposed methods for discovering holes [25, 26], and we propose an algorithm of generating holes detectors with variable-sized r-chunk matching based on the discovered hole set.

Define self set $S = \{s_1, s_2, \cdots, s_N\}$ which has N elements. Define hole set $H = \{h_1, h_2, \cdots, h_M\}$ which has M elements. Define the length of element strings is L. Self set is represented by a matrix as follows:

$$S = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,i} & \cdots & s_{1,L} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,i} & \cdots & s_{2,L} \\ \vdots & & & & & \\ s_{N,1} & s_{N,2} & \cdots & s_{N,i} & \cdots & s_{N,L} \end{bmatrix} \quad (1)$$

According to r-contiguous bits, the self set S is divided into $L - r + 1$ sets by the r-step sliding window, shown as follows:

$$S = [S_r[1], S_r[2], \cdots, S_r[i], \cdots, S_r[L-r+1]] \quad (2)$$

$$S_r[i] = \left\{ \begin{matrix} s_{1,i} & \cdots & s_{1,i+r-1} \\ s_{2,i} & \cdots & s_{2,i+r-1} \\ \vdots & s_{j,i} & \vdots \\ s_{N,i} & \cdots & s_{N,i+r-1} \end{matrix} \right\}, i = 1, 2, \cdots, L-r+1 \quad (3)$$

Similarly, the hole set H can also be divided into $L - r + 1$ sets.

$$H = [H_r[1], H_r[2], \cdots, H_r[i], \cdots, H_r[L-r+1]] \quad (4)$$

$$H_r[i] = \left\{ \begin{matrix} h_{1,i} & \cdots & h_{1,i+r-1} \\ h_{2,i} & \cdots & h_{2,i+r-1} \\ \vdots & h_{j,i} & \\ h_{M,i} & \cdots & h_{M,i+r-1} \end{matrix} \right\}, i = 1, 2, \cdots, L-r+1 \quad (5)$$

Calculating the difference set $H_r[i] - S_r[i]$. If exist $H_r[i](j) = h_{j,i}h_{j,i+1} \cdots h_{j,i+r-1}$ that does not appear in the self set $S_r[i]$, then $H_r[i](j)$ can generate r-chunk matching detector $d_{r,i} = \{i, h_j\}$ to detect the hole h_j.

The hole detectors generation algorithm based on variable-sized r-chunk is described as follows:

- Step 1: Define self set S and element length L;
- Step 2: Calculate the hole set H according to the EHANDP algorithm [26];
- Step 3: According to the step r of the sliding window, divide S and H into $L - r + 1$ sets respectively, and the initial r-chunk matching length is $r = r_0$.
- Step 4: Calculate $D_r[i] = H_r[i] - S_r[i]$, $i = 1, 2 \cdots L - r + 1$. For the non-empty set $D_r[i]$, generate the holes detector set $D_r = \{d_{r,i}\}$ and delete the holes that can be covered by D_r from H.
- Step 5: Determine whether there are still some holes in H, and if so, increase the matching length r by 1 and the value of r is $r_0, r_1 \cdots$ to maximum L in turn, then go to Step 3; Otherwise, the set of hole detectors $D_H = D_{r_0} \cup D_{r_1} \cup \cdots$ can cover all holes, and the algorithm ends.

The above algorithm is exemplified as follows.

Given a self set S = {01001, 00001, 10100, 11101} with r-contiguous bits matching where r = 3 and L = 5. Calculate the hole set H = {00000, 01000, 10101, 11100} according to the EHANDP algorithm. Let the initial r-chunk matching length $r_0 = 3$, and divide S and H into $L - r_0 + 1 = 3$ sets as follows:

$$S_3[1] = \begin{bmatrix} 010 \\ 000 \\ 101 \\ 111 \end{bmatrix}, S_3[2] = \begin{bmatrix} 100 \\ 000 \\ 010 \\ 110 \end{bmatrix}, S_3[3] = \begin{bmatrix} 001 \\ 001 \\ 100 \\ 101 \end{bmatrix} \tag{6}$$

$$H_3[1] = \begin{bmatrix} 000 \\ 010 \\ 101 \\ 111 \end{bmatrix}, H_3[2] = \begin{bmatrix} 000 \\ 100 \\ 010 \\ 110 \end{bmatrix}, H_3[3] = \begin{bmatrix} 000 \\ 000 \\ 101 \\ 100 \end{bmatrix} \tag{7}$$

$$H_3[1] - S_3[1] = \emptyset \tag{8}$$

$$H_3[2] - S_3[2] = \emptyset \tag{9}$$

$$H_3[3] - S_3[3] = \{000\} \tag{10}$$

$H_3[3] - S_3[3]$ can generate r-chunk detector $d = \{3{,}00000\}$, which can detect holes 00000 and 01000.

Then, there are still two holes in $H = \{10101, 11100\}$. In order to further detect the remaining holes, let $r_1 = r_0 + 1 = 4$, divide S and H into $L - r_1 + 1 = 2$ sets.

$$S_4[1] = \begin{bmatrix} 0100 \\ 0000 \\ 1010 \\ 1110 \end{bmatrix}, S_4[2] = \begin{bmatrix} 1001 \\ 0001 \\ 0100 \\ 1101 \end{bmatrix} \tag{11}$$

$$H_4[1] = \begin{bmatrix} 1010 \\ 1110 \end{bmatrix}, H_4[2] = \begin{bmatrix} 0101 \\ 1100 \end{bmatrix} \tag{12}$$

$$H_4[1] - S_4[1] = \varnothing \tag{13}$$

$$H_4[2] - S_4[2] = \{0101, 1100\} \tag{14}$$

$H_4[2] - S_4[2]$ can generate detectors $d = \{2{,}10101\}$ and $d = \{2{,}11100\}$ with matching length $r_1 = 4$, which can detect holes 10101 and 11100. So far all the holes in H can be detected.

5 Ransomware Analysis and Feature Extraction

5.1 API Call Sequences of Ransomware

The behavior features of ransomware are extracted in a virtual environment named Cuckoo Sandbox that automated the task of analyzing malicious file [13]. In our virtual analyzing environment, some types of fishing files (e.g., jpg, doc, xls, pdf, sql, etc.) are intentionally placed in different directories to trigger the encryption operation of ransomware. Most of the ransomware's implementation of cryptographic algorithm (e.g., RSA and AES) is depended on CryptoAPI that is included in Windows [27].

According to the function of different system calls, we define sixteen categories of API functions. We list some API functions examples of six typical categories that are chosen from the total sixteen categories, shown as Table 1.

Table 1. Categories of API functions

Categories	Description	Examples of API functions
Crypto	Encryption and decryption of data	CryptEncrypt CryptDecrypt CryptHashData
File	File operations, such as read, write, delete, et al.	Writefile MoveFileEx DeleteFile
Process	Process and thread operations	NtOpenProcess NtAllocateVirtualMemory NtTerminateProcess
Service	Service operations, such as create, start and stop	OpenSCManager OpenService StartService
System	System operations	LdrLoadDll NtQuerySystemInformation SetWindowsHookEx
Misc	Other miscellaneous operations	GetComputerName GetUserName GetTimeZoneInformation

5.2 *n*-gram Feature Selection

To extract features from API call sequences the *n*-gram model is used. API function names are treated as words of *n*-gram, and extract the *n*-gram features from both the ransomware samples and benign samples [16].

Information gain (*IG*) is used as a feature selection method. Let x be the *n*-gram feature, $y_i \in y$ be one of the k class sample labels (i.e., $k = 2$, ransomware or benign), and the $IG(x)$ be the *IG* weight for feature x, calculated as follows [16].

$$\begin{aligned} IG(x) &= H(y) - H(y|x) \\ &= -\sum_{i=1}^{k} p(y_i)\log(p(y_i)) \\ &\quad + p(x)\sum_{i=1}^{k} p(y_i|x)\log(p(y_i|x)) \\ &\quad + p(\bar{x})\sum_{i=1}^{k} p(y_i|\bar{x})\log(p(y_i|\bar{x})) \end{aligned} \tag{15}$$

Compute the *IG* value for each *n*-gram feature, and the larger the *IG* value is, the more information the feature contributes for classifying ransomware and benign samples. The feature set $F = \{f_1, f_2, \cdots, f_N\}$ is composed of the top N features based on *IG* values.

5.3 Feature Vector Encoding

Based on the feature set $F = \{f_1, f_2, \cdots, f_N\}$, each sample will be encoded to a feature vector $v = (v_1, v_2, \cdots, v_N)$, where v_i represents whether the *n*-gram feature f_i appears in the API call sequences of the sample. If f_i appears then v_i is 1, otherwise v_i is 0.

The feature vectors are treated as binary strings in the DL-NSA. The training benign samples are encoded to self set S, the training ransomware samples are encoded to antigens which are used for generating mature detector set D. The test samples (both benign and ransomware) are encoded to feature vectors and then be distinguished by the detectors of DL-NSA.

6 Experimental Design

6.1 Collation of Ransomware and Benign Samples

We collect 2,000 ransomware samples of the popular ransomware families, such as CryptoWall, Wannacry, Cerber, etc. We also gather 1,000 benign samples. These two types of samples are both equally divided into two collections: training samples collection is used for immune detectors generation, and test samples collection is used for validating the detection effect.

6.2 Experimental Results and Analysis

Distribution of Hole Detectors

The training samples collection contains 500 benign samples, and these samples are analyzed in Cuckoo sandbox. All the benign samples are encoded to feature vectors $v = (v_1, v_2, \cdots, v_N)$ which compose the self set, and in our experiment let $N = 30$. For the first layer of DL-NSA using *r*-contiguous bits matching with the matching length *r*, we can obtain all the holes. Then, according to the hole detector generation algorithm we proposed, the second layer variable-sized *r*-chunk hole detectors of DL-NSA are generated. The distribution of different matching threshold of hole detectors is shown as Fig. 5.

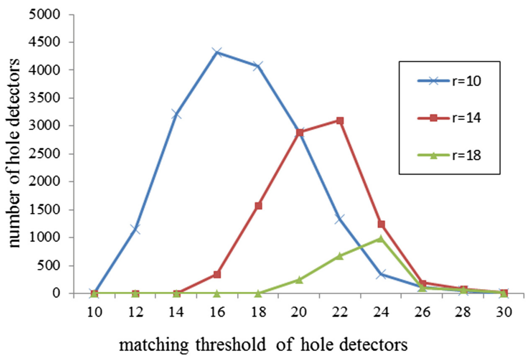

Fig. 5. Distribution of hole detectors

We choose three matching length values (i.e. r = 10, 14, 18) of the first layer mature detectors. The number of hole detectors is first increased and then decreased along with the increasing of the matching threshold of hole detectors. With the increase of *r*, the total number of hole detectors is decreased, and the peak of the curves shift right. This experiment indicates that the hole detectors of the second layer are affected by the matching length of detectors of the first layer.

Detectors' Proportion of the Two Layers Affects the Detection Rate
The non-self space is divided into two parts: detectable space and hole space according to whether the elements in non-self space can be detected by the *r*-contiguous bits matching of the first layer. Non-self space coverage includes two aspects: detectable space coverage and hole space coverage. In this part, we will analysis the detectors' proportion of the two layers how to affect the detection rate.

First, we build a non-self space dataset of 2,000 elements, which are the feature vectors extracted from ransomware or random generated. The non-self space dataset contains 1,000 detectable elements and 1,000 hole elements.

The total number of detectors of DL-NSA is *N*, including mature detectors of the first layer and hole detectors of the second layer. Define the percentage of hole detector is p. For the first layer, *r*-contiguous bits matching is used with the matching length is 10, and the number of mature detectors is $N_1 = N * (1 - p)$. For the second layer, variable-sized *r*-chunk matching is used, with the initial matching length r = 10, and the number of hole detectors is $N_2 = N * p$. If $N_2 > N_H$, that means N_2 is greater than the actual total number of holes, then let $N_2 = N_H$ and $N_1 = N - N_H$.

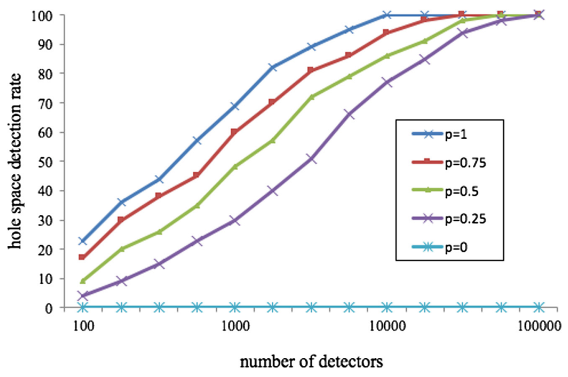

Fig. 6. Hole space detection rate

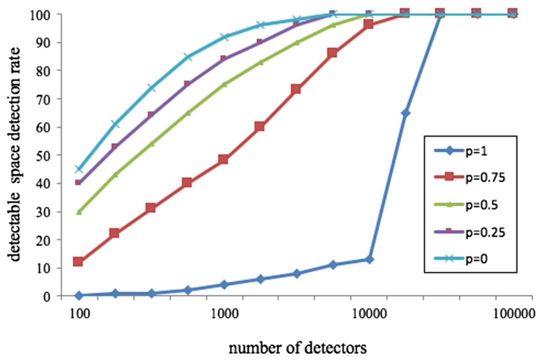

Fig. 7. Detectable space detection rate

As shown in Figs. 6 and 7, the detection rate of hole space and detectable space both increase gradually with the increase of the total number of detectors. When the total number of detectors is fixed, the larger p is, the higher detection rate for the hole space and the lower detection rate for the detectable space.

When $p = 0$, DL-NSA degenerates to the original NSA, and the system has a high detection rate for the detectable space, but the hole space detection rate is 0.

When $p = 1$, the first layer detector is generated only when the actual maximum number of hole detectors reaches. As can be seen from Fig. 7, when $p = 1$, the detection rate of detectable space increases slowly in the first half, because only the hole detectors are generated which can cover a small area of detectable space. When $N > N_H$, the detection rate of detectable space increases rapidly.

Ransomware Detection

This section compares the ransomware detection performance of NSA proposed by Forrest and r-adjustable negative selection algorithm (RA-NSA) [23] with DL-NSA proposed in this paper.

For NSA algorithm, the matching length of detectors is set to 10 and the number of detectors is N.

For RA-NSA algorithm, the initial matching length of detector is set to 10, the upper limit of matching length is 25, and the number of detectors is N.

For DL-NSA algorithm, the total number of detectors is N, and the percentage of hole detectors is p. For DL-NSA algorithm, in order to ensure high performance in both detectable space and hole space, $p = 0.1$ is chosen.

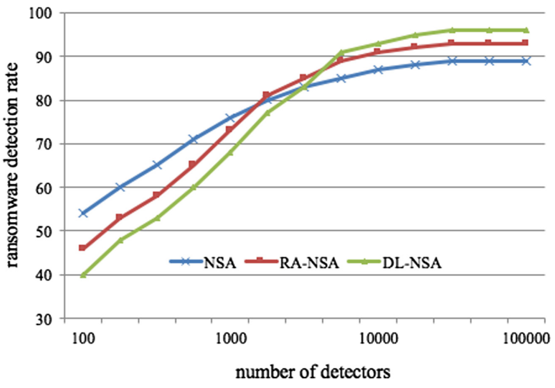

Fig. 8. Ransomware detection

As shown in Fig. 8, the detection rate of ransomware shows an upward trend, especially in the initial stage. When the number of detectors is greater than 10,000, the detection rates of the three algorithms tend to be stable. When the number of detectors is small, the coverage space of detectors in NSA algorithm is the largest, so the detection effect of NSA is the best. When the number of detectors is large, DL-NSA algorithm has better coverage of non-self space, especially for hole elements, so the detection effect is the best, up to 96%.

In the practical application of ransomware defense, it is suggested that the combination of NSA and DL-NSA should be used, determined by system requirement of the detection speed and accuracy. If high detection speed is demanded, fewer detectors should be used to achieve more non-self space coverage. At this time, the use of NSA before the intersection of two curves (NSA and DL-NSA) can guarantee the speed while having a higher detection rate. If high detection accuracy is demanded, more detectors should be used, especially to cover the hole space better. At this time, using DL-NSA after the intersection of two curves (NSA and DL-NSA) can achieve a higher detection rate of ransomware.

7 Conclusion

Ransomware has attracted wide attention from government, security companies and scientific researchers. In order to effectively identify ransomware, the main work of this paper is as follows.

(1) An API call sequences feature extraction and selection method of ransomware is proposed. The Cuckoo sandbox is used for monitoring the sensitive behaviors such as file reading and writing, data encryption and so on, during the execution of the ransomware.
(2) A method of generating hole detectors with variable-sized *r*-chunk matching is proposed. The original NSA is improved, and a DL-NSA model is constructed by introducing a double-layer detectors.
(3) The improved DL-NSA is applied to the detection of ransomware. By adjusting the proportion of two-layer detectors, better ransomware detection effect of ransomware can be achieved.

Although some ransomware recovery and decryption tools have been released by security companies, most ransomware cannot be cracked in fact. Therefore, in the defense of ransomware, the prevention in advance and file backup are more important. The confrontation between security companies and ransomware writers will continue. The application of cryptographic algorithms will make the confrontation more intense. Higher-strength cryptographic algorithms and more complex combinations of cryptographic algorithms will appear. So we must get prepared, and put forward the method of analysis, detection and protection for ransomware.

Acknowledgments. This work was supported by the National Key R&D Program of China (2016YFB0801100), the National Natural Science Foundation of China (61602489), the Fundamental Research Funds for the Central Universities of PPSUC (2019JKF108) and the National Cryptography Development Fund (MMJJ20180108).

References

1. Muhammad, U.K., Jantan, A.: The age of ransomware: understanding ransomware and its countermeasures. In: Artificial Intelligence and Security Challenges in Emerging Networks, pp. 1–4. IGI Global, Pennsylvania (2019)
2. Masarah, P.C., Bernhard, H., Benoit, D.: Ransomware payments in the bitcoin ecosystem. In: Proceeding of the 17th Annual Workshop on the Economics of Information Security (WEIS), pp. 1–10. Innsbruck (2018)
3. Rehman, H., Yafi, E., Nazir, M., Mustafa, K.: Security assurance against cybercrime Ransomware. In: Vasant, P., Zelinka, I., Weber, G.-W. (eds.) ICO 2018. AISC, vol. 866, pp. 21–34. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-00979-3_3
4. Maigida, A.M., Abdulhamid, S.M., Olalere, M., et al.: Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms. J. Reliable Intell. Environ. **5**(2), 67–89 (2019)
5. Hull, G., John, H., Arief, B.: Ransomware deployment methods and analysis: views from a predictive model and human responses. Crime Sci. **8**(1), 1–22 (2019)

6. Santos, I., Brezo, F., Ugarte-Pedrero, X., et al.: Opcode sequences as representation of executables for data-mining-based unknown malware detection. Inf. Sci. **231**(9), 203–216 (2013)
7. Wang, T., Xu, N.: Malware variants detection based on opcode image recognition in small training set. In: Proceedings of the 2nd IEEE International Conference on Cloud Computing and Big Data Analysis, pp. 328–332. IEEE, Piscataway (2017)
8. Zhang, H., Xiao, X., Mercaldo, F.: Classification of ransomware families with machine learning based on n-gram of opcodes. Future Gener. Comput. Syst. **90**(2019), 211–221 (2019)
9. Sgandurra, D., Muñoz-González, L., Mohsen, R., et al.: Automated dynamic analysis of ransomware: benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020. Accessed 1 December 2016
10. Xu, Z., Ray, S., Subramanyan, P., et al.: Malware detection using machine learning based analysis of virtual memory access patterns. In: Proceedings of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 169–174. IEEE, Piscataway (2017)
11. Scaife, N., Carter, H., Traynor, P., et al.: CryptoLock (and drop it): stopping ransomware attacks on user data. In: Proceedings of the 36th International Conference on Distributed Computing Systems, pp. 303–312. IEEE, Piscataway (2016)
12. Hampton, N., Baig, Z., Zeadally, S.: Ransomware behavioural analysis on windows platforms. J. Inf. Secur. Appl. **40**(2018), 44–51 (2018)
13. Lu, T.L., Zhang, L., Wang, S.Y., et al.: Ransomware detection based on V-detector negative selection algorithm. In: Proceedings of the 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), pp. 531–536. IEEE, Piscataway (2017)
14. Gao, X.Z., Chow, M.Y., Pelta, D., et al.: Theory and applications of artificial immune systems. Neural Comput. Appl. **19**(8), 1101–1102 (2010)
15. Dasgupta, D., Yu, S., Nino, F.: Recent advances in artificial immune systems: models and applications. Appl. Soft Comput. **11**(2011), 1574–1587 (2011)
16. Lu, T.L., Zhang, L., Fu, Y.X.: A novel immune-inspired shellcode detection algorithm based on hyper-ellipsoid detectors. Secur. Commun. Netw. **8**(2018), 1–10 (2018)
17. Tan, Y.: Artificial Immune System: Applications in Computer Security. IEEE Computer Society Press, Piscataway (2016)
18. Hooks, D., Yuan, X., Roy, K., et al.: Applying artificial immune system for intrusion detection. In: Proceedings of IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), pp. 287–292. IEEE, Piscataway (2018)
19. Brown, J., Anwar, M., Dozier, G.: Detection of mobile malware: an artificial immunity approach. In: Proceedings of 2016 IEEE Security and Privacy Workshops (SPW), pp. 74–80. IEEE, Piscataway (2016)
20. Iqbal, M., Abid, M.M., Ahmad, M.: Catching Webspam Traffic with Artificial Immune System (AIS) classification algorithm. In: Proceedings of the 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 402–405. IEEE, Piscataway (2017)
21. Forrest, S., Perelson, A.S., Allen, L., et al.: Self-nonself discrimination in a computer. In: Proceedings of 1994 IEEE Symposium on Research in Security and Privacy, pp. 202–212. IEEE, Piscataway (1994)
22. Hofmeyr, S.A.: An immunological model of distributed detection and its application to computer security. Department of Computer Sciences, University of New Mexico (1999)
23. Zhang, H., Wu, L.F., Zhang, R.S., et al.: An algorithm of r-adjustable negative selection algorithm and its simulation analysis. Chin. J. Comput. **28**(10), 1614–1619 (2005)
24. Ji, Z., Dasgupta, D.: Revisiting negative selection algorithms. Evol. Comput. **5**(2), 223–251 (2007)

25. Stibor, T., Mohr, P., Timmis, J.: Is negative selection appropriate for anomaly detection. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 321–328. ACM, New York (2005)
26. Liu, X.B., Cai, Z.X.: Properties assessments of holes in anomaly detection systems. J. Cent. South Univ. (Sci. Technol.) **40**(4), 986–992 (2009)
27. Kirda E.: UNVEIL: a large-scale, automated approach to detecting ransomware (Keynote). In: Proceedings of IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), p. 1. IEEE, Piscataway (2017)