Improvement on Anomaly Intrusion Detection Algorithm Based on V-Gram

Jifeng Yu and Jingli Zhou

Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei Province, China

Abstract

Short sequences of system calls of running processes are good data sets for anomaly detection system. This paper analyzed the insufficiency of fixed-length patterns (N-Gram) and variable-length patterns (V-Gram) in processing sequences of system calls. A new model of Multi Wildcards V-Gram (MWV-Gram) with redundancy controlling mechanism is presented. Experimentation results indicate that the pattern database is reduced and detection efficiency is enhanced by the improved algorithm.

Keywords: Anomaly detection, V-Gram, Sequence pattern.

1. Introduction

With the rapid development and popularity of computer networks, the number of malware attacks is increasing quickly. Traditional intrusion detection systems based on signature and pattern matching are efficient for known attacks but suffer from novel attacks. Anomaly detection systems model normal behavior of applications. Any behaviors that deviate from normal model are considered abnormal. Anomaly detection system can detect unknown attacks.

In intrusion detection systems, sequences can be considered event streams (e.g., system call sequences, access request sequences). The chronological relationship of commands executed by user or program can be used for anomaly detection [1, 2, 3]. First, we mine those sequences that appear frequently, namely frequent sequences. Assume \( P \) is a sequence, if there are \( s \) sequences in training sequences that contain \( P \), and then the support of \( P \) is \( s \). Given a threshold \( SUP \), if \( s > SUP \), then \( P \) is a frequent sequence. These frequent sequences compose standard pattern database. Anomalous sequence does not match any sequence in standard pattern database.

Forrest [4] proposed a way of intrusion detection using system call sequences of applications. Fixed-length frequent system call sequences are extracted to compose standard pattern database. The database
covers the basic behavior of the application. This model is called N-Gram [5] model. It uses a sliding window with size \( N \) to operate on the training sequences to form sequence fragments of length \( N \). Then computes the frequency of their occurrence statistics for all fragments. Set a threshold \( SUP \), all sequences below this threshold will be discarded. The remaining sequences compose standard pattern database. In detection phase, use the same window with size \( N \) to slide on sequence to be tested. For each sequence obtained of length \( N \), search it in standard pattern database. If found, call it match, otherwise known as mismatch.

A further development of this idea was proposed by Wespi[6,7]. Instead of fixed-length patterns, the standard pattern database consists of variable-length patterns. This model is called V-Gram model. It uses TEIRESIAS [7, 8] algorithm to generate variable-length sequences. This algorithm is divided into two stages: scanning and convolution. In scanning stage, it generates all the variable-length patterns which have sufficient support. In convolution stage, restitches together the sequence fragments to generate largest pattern set. The amount of patterns in standard pattern database of V-Gram is less than that of N-Gram. In general, applications generate different system call sequences in different scenarios, and the length of system call sequences is different. So V-Gram describes the behavior of application better.

This paper is organized as follows. First, an improved MWV-Gram algorithm is introduced in Section 2. Then, in Section 3, we describe the flow of MWV-Gram algorithm. Section 4 presents the experiment results. Section 5 summarizes with some concluding remarks.

2. MWV-Gram model

In N-Gram model, it’s difficult to determine the value of \( N \). If \( N \) is too small, the amount of frequent sequences is big and anomaly signal is too weak to detect. On the other hand, too big \( N \) makes the pattern too rigid and results in high false positive rate. V-Gram model is more flexible than N-Gram model, but it still has shortcoming. Usually the application code consists of three basic structures: sequence, select ion and loop. For sequence structure, V-Gram model is efficient for system call sequence detection. But for loop structure, N-Gram and V-Gram models can’t express them well. In order to express loop structure better, we improved the algorithm based on V-Gram. A new wildcard is introduced in this paper. We call this new model Multi Wildcards V-Gram (MWV-Gram).

To illustrate the MWV-Gram model, we define several concepts here.

**Pattern:** A regular expression is a pattern. If pattern \( A \) is shorter than pattern \( B \), and is part of pattern \( B \), then \( A \) is sub-pattern of \( B \). We use character \( \cdot \) to represent any character in a pattern.

**<L,W> pattern:** There are two parameters(\( L \) and \( W \)) in this pattern and \( L \leq W \). If pattern \( P \) is \(<L,W> \) pattern, then any sub-pattern of \( P \) whose length is greater than or equal to \( W \) (including \( \cdot \) ) has at least \( L \) elements (not include \( \cdot \)).

**Prefix and Suffix of pattern:** These patterns are special sub-patterns because they are at the beginning or end of a pattern. Assume that pattern \( P \) contains at least \( L \) elements, \( \text{prefix}(P) \) is a prefix of \( P \) and contains \( L-1 \) elements. \( \text{suffix}(P) \) is the suffix of pattern \( P \) and contains \( L-1 \) elements. For example, let \( L=3 \), then:

\[
\text{prefix}(‘W.CBACD’) = ‘W.C’, \quad \text{prefix}(‘CDBAC’) = ‘CD’
\]

\[
\text{suffix}(‘W.BAC.D’) = ‘C.D’, \quad \text{suffix}(‘BAC..D’) = ‘C..D’
\]

**Convolution:** Let pattern \( P, Q \) be arbitrary patterns with at least \( L \) elements; pattern \( Q’ \) is the remaining part of \( Q \) after the \( \text{prefix}(Q) \) is removed; the convolution of \( P \) and \( Q \) is a new pattern \( R \) defined as follows:

\[
R = P \oplus Q = \begin{cases} P Q’ & \text{if } \text{suffix}(P) = \text{prefix}(Q) \\ \text{NULL} & \text{otherwise} \end{cases}
\]
Specific pattern: If pattern $Q$ can be obtained from $P$ by extending or specialization, we call $Q$ is more specific than $P$. Extending means appending new elements or wildcards to pattern $P$. Specialization means changing one or more wildcards to elements. The following patterns are more specific than pattern ‘D.AH..C’:


Obviously, the support of $Q$ is not more than $P$.

Maximum Pattern: Given a sequence set $S$, a pattern $P$ is called maximal with respect to $S$ if there exists no pattern $P'$ which is more specific than $P$. In other words, the supports of all specific patterns of $P$ are less than the support of $P$.

Redundant pattern: Let $P$ be the maximum pattern, $P_1$, $P_2$, $P_3$, ... $P_n$ are the specific patterns of $P$. If each appearance of $P$ in sequence set can be replaced by one of the sequence $P_1$, $P_2$, $P_3$, ... $P_n$, we call $P$ redundant pattern.

Suppose sequence set $S=${$CDEAFBXY$, $CAEBXFYH$, $CDBAEBXY$, $CAJBXHGK$}, Table 1 lists some of the maximum patterns and redundant patterns.

```
<table>
<thead>
<tr>
<th>Maximum pattern</th>
<th>Support</th>
<th>Redundant pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.BX</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CA.BX</td>
<td>2</td>
<td>A.BX</td>
</tr>
<tr>
<td>A.BXY</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
```

From Table 1, we can see that pattern $A.BX$ is maximum pattern because the support of any more specific pattern is lower than it. But pattern $CA.BX$ and $A.BXY$ can cover each appearance of pattern $A.BX$ in sequence set $S$. So pattern $A.BX$ is redundant pattern with respect to the other two patterns.

Our new MWV-Gram model needs to meet the following requirements:

1. The length of pattern is not fixed.
2. Pattern is $<$L, W$>$ pattern.
3. Pattern is frequent pattern.
4. Pattern is maximum pattern.
5. Pattern is not redundant pattern.

We use the following pseudo code which contains sequence, selection and loop structures to compare the three models.

```java
C;
while(
{|D;
|if() E;
|else if() F;
|else() G;
|H;
|G;
|H;
```

Where C,D,E,F,G, H are system calls. Table 2 lists the results of the three algorithms modeling on the code above. For N-Gram (N=3) model, there needs 17 patterns to cover all the normal patterns; V-Gram model needs 8 patterns. Because MWV-Gram model can identify the loop structure, there needs only two patterns to cover the behavior of the code. Three models can describe sequence structure, V-Gram and MWV-Gram can use wildcard to describe selection structure. As for MWV-Gram, we present several new wildcards to describe loop structure. Similar to regular expression, we use ‘( )’ to represent
the loop body, and ‘+’ to represent that the loop on the left occurs one or more times. Obviously, MWV-Gram model is the most direct expression of the three basic code structures.

### TABLE 2  Comparison of three models

<table>
<thead>
<tr>
<th>System call sequence</th>
<th>N-Gram(N=3)</th>
<th>V-Gram</th>
<th>MWV-Gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGH</td>
<td>CGH CDE</td>
<td>CGH</td>
<td>CGH C(D.H)+GH</td>
</tr>
<tr>
<td>CDEGH</td>
<td>DEH EHG</td>
<td>CDEH</td>
<td></td>
</tr>
<tr>
<td>CDGH</td>
<td>CDG DGH</td>
<td>CDG</td>
<td></td>
</tr>
<tr>
<td>CDEHDFGH</td>
<td>DFH FHG</td>
<td>CDFH</td>
<td></td>
</tr>
<tr>
<td>CDDEHDFGH</td>
<td>CDG DGH</td>
<td>CDEH</td>
<td></td>
</tr>
<tr>
<td>CDGHDGFH</td>
<td>GHG EHD</td>
<td>CDG</td>
<td></td>
</tr>
<tr>
<td>CDGHDGFH</td>
<td>HDF HDG</td>
<td>DEH</td>
<td></td>
</tr>
<tr>
<td>CDGHDGFH</td>
<td>FHD HDE</td>
<td>DFH</td>
<td></td>
</tr>
<tr>
<td>CDEHDFHDG</td>
<td>GHD</td>
<td>DGH</td>
<td></td>
</tr>
</tbody>
</table>

Loop structure is a basic structure of the program, and it will produce a large number of repeated sequences of system calls. These repeated sequences do not produce new pattern, but result in tremendous training and detection overhead. Suppose the percentage of mismatched sequences is $t$, $t$ is called anomaly index. Given a threshold $SUP$, it is determined that an attack has occurred when $t > SUP$. If a sequence has many repeated fragments, and the anomaly fragments is not in the loop structure, most likely the anomaly index is too weak to detect. Therefore, eliminating repeated sequences not only reduces the overhead of training and detection, but also improves the detection rate.

### 3. Description of MWV-Gram algorithm

MWV-Gram algorithm is divided into three stages: first we need to recognize and eliminate the loop fragments in training sequence, then generate all frequent $<L, W>$ patterns. The last step is to convolute on these $<L, W>$ patterns and produce MWV-Gram patterns.

#### 3.1 Recognizing and eliminating loops

Identify loop fragments is simple. For example, using the code above, we can generate the following sequence:

$$\text{CDEHDFHDFHDEHDEHDGDGDGHDFHG}$$

There are three loop fragments in the sequence:  

$$DFH, DEH, DGH.$$  

After the elimination of loop fragments, the sequence is as follows:  

$$\text{CDEHDFHDEHDEHDGDGDGHDFHG}$$

We need to mark the positions of the eliminated loop fragments to generate loop patterns later. Elimination of loop fragments will not cause information loss of the sequence, and can reduce the impact of large amount of redundant information in the sequence.

#### 3.2 Generating $<L, W>$ pattern

After the elimination of loop fragments, the algorithm generates all the $<L, W>$ patterns with the length L. Pratt[9] algorithm is based on the exhaustive depth-first algorithm, and it can generate all the frequent $<L, W>$ patterns, these patterns is called basic patterns. There is a pattern tree in the algorithm. Each branch of the tree represents a frequent $<L, W>$ pattern. Extending from the known short patterns
until the support of pattern is lower than the threshold predefined, so that generates a new $<L, W>$ pattern. Then extending from the previous branch until all the $<L, W>$ patterns are mined. This algorithm can search out longer and more complex patterns than simple exhaustive algorithm, but it generates a lot of temp patterns at expanding stage. So the scanning time is increased and need to calculate the support of these temp patterns. We made some improvement to Pratt algorithm. We generate a separate pattern tree for each pattern with the length between $L$ and $W$, so the number of temp patterns is reduced. We only need to calculate the support of patterns which actually exist, and the number of scanning is reduced. Let $N=W-L$, represents the number of wildcards basic pattern can contain. $P$ is the pattern tree to be constructed. The length of $P$ is $M$, and support threshold is $SUP$. The algorithm is as follows:

- Establish the root node of the tree.
- Use a sliding window with size $M$ to operate on the training sequences to form sequence fragments of length $M$, and calculate its support $s$, if $s>SUP$, then a fixed length frequent sequence $Q$ is generated.
- Matching $Q$ with all the branches of pattern tree $P$, calculating the mismatch number $U$ of each branch, finding the branch $Q'$ whose mismatch number $U$ is the smallest. If $U<=$N, merge $Q$ into the $Q'$ branch and replace the mismatch position with a wildcard. If $U>N$, establish a new branch from the root node.
- Continue to slide the window until traverse all sequences.

### 3.3 Convolution

TEIRESIAS algorithm uses the basic patterns generated in scanning stage as input to combine them using convolution operation. The following is the basic steps:

- Put all the basic patterns into the stack according certain rules.
- Suppose the pattern at the top of the stack is $P$; we call $P$ current top pattern.
- Find all the patterns $Q$ in the stack which are convolvable with $P$, and extend pattern $P$ to the ‘right'(suffix) direction by convolution with $Q$. Let $R$ be the convolution result, if the support of $R$ is lower than threshold $SUP$ predefined, $R$ is discarded; otherwise, place $R$ at the top of the stack. Then start this step recursively until the current top pattern can not be extended.
- Similar to step 3, this time extend $P$ to the left(prefix) direction.
- When the extension in both directions has been completed, pop the current top pattern, if it is maximum pattern, save it to pattern database. Repeat the above steps until all the patterns in stack have been processed.

The problem with this algorithm is: There is not redundant control mechanisms during the extending procedure, maybe some of the output $<L, W>$ patterns are redundant pattern. Even if a maximum pattern can not be covered by a more specific pattern, if may be covered by a group of more specific patterns, and that would result in redundancy. We made some modification to TEIRESIAS algorithm. Because more specific pattern is generated from not more specific patterns by convolution, here we introduce a new offset list which records the match state of current top pattern in training sequence.

$$List(P) = \{(No\_P_i, Pos\_P_i, Over\_P_i) | i = 1, 2, ..., N\}$$

Where $P$ represents the current top pattern, $No\_P_i$ is the sequence number, $Pos\_P_i$ is the position pattern $P$ occurs, $Over\_P_i$ represents whether this pattern is covered by another more specific pattern.

Use the system call sequence in Table 2 as example, pattern $P='DE'$, the offset list of P is:

$$\{(2,2,0),(5,2,0),(6,2,0),(8,2,0),(8,5,0),(11,2,0)\}$$

Checking the offset list of current top pattern before every convolution, if all $Over\_P_i$ in list are 1, then current pattern is redundant pattern, stop current process and operate on the next pattern. The value of $Over\_P_i$ is updated after each convolution.
The convolution of MWV-Gram algorithm uses the basic patterns generated in section 3.2 as input to generate non-redundant MWV-Gram patterns with redundant control mechanisms. Checking the patterns generated, if it contains the loop mark described in section 3.1, changes the loop fragment into ( )+ style. Because the patterns generated by MWV-Gram algorithm contain multi wildcards, this algorithm can describe the selection and loop structure well.

4. Experiments

In our experiments, we generate patterns using these three pattern generation algorithms, compare the size of pattern database and detection overhead. We use the same CERT sendmail data set as employed by Forrest, et al. [10] at the University of New Mexico in our experiments. This data set contains normal and intrusion data, including a variety of system call sequence scenarios. Each scenario corresponds to a sequence set. We established pattern database for each normal scenario, the size of database is shown in Table 3. Set N=6 for N-Gram; L=3, W=4 for MWV-Gram.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>N-Gram</th>
<th>V-Gram</th>
<th>MWV-Gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>sendmail.daemon</td>
<td>217</td>
<td>61</td>
<td>44</td>
</tr>
<tr>
<td>sendmail</td>
<td>71</td>
<td>32</td>
<td>27</td>
</tr>
</tbody>
</table>

As can be seen, the size of pattern database of MWV-Gram is the smallest because it can recognize loop fragment and the redundant patterns are removed.

<table>
<thead>
<tr>
<th>Detection Overhead(ms/trace)</th>
<th>N-Gram</th>
<th>V-Gram</th>
<th>MWV-Gram</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>34.6</td>
<td>41.1</td>
<td>7.9</td>
</tr>
</tbody>
</table>

Table 4 records the detection overhead of the three models. First, calculate the time three models spent for detecting all the sequences, and then calculate the average time spent for detecting each sequence. Although MWV-Gram model is more complex than the other two models and spends more time in training stage, yet the pattern database generated after training is smaller, and detection efficiency is much higher than the other two models.

5. Concluding Remarks

In this paper, the application system call sequence is used as anomaly detection data sources. We analyze three algorithms which are N-Gram model, V-Gram model and MWV-Gram model. The experiment shows that MWV-Gram model can describe the sequence, selection and loop structure well with a smaller pattern database and very low detection overhead.

References