© 2015 IJSRSET | Volume 1 | Issue 3 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Hybrid Algorithm for Backward Hashing and Automation Tracking For Virus Scanning

Panchal Mital K, Bakul Panchal
Department of Information Technology, L. D. College Of Engineering, Ahmedabad, Gujarat, India

ABSTRACT

Virus scanning involves computationally intensive string matching against a large number of signatures of different characteristics. Matching a variety of signatures challenges the selection of matching algorithms. We propose a hybrid approach that partitions the signatures into long and short ones in the open-source ClamAV for virus scanning. By improving and enhancing the Wu-Manber algorithm, the new algorithm called as the Backward Hashing algorithm which is dependable for only long patterns to extend the average skip distance. There is one more algorithm which takes care of short patterns is Aho-Corasick algorithm. It scans only short patterns to reduce the automation sizes. Algorithm we have discussed first uses the bad-block heuristic to develop long shift distance and thus decreasing the verification rate of recurrence. In that way it is much faster than the original WM implementation in ClamAV open source antivirus software. Algorithm we have stated later increases the AC performance by around 50 percent due to better cache locality. We also rank the factors to indicate their importance for the string matching performance.

Keywords: ClamAv, AC, Backward Hashing

I. INTRODUCTION

Using two or more algorithms as base and adding new viable features to it, we have created the proposed hybrid algorithm. New proposed algorithm come into with some properties of its base paper algorithms and performs operations better. Proposed new (hybrid) algorithms are proposed for both exact string matching and approximate string matching. Before our proposed algorithm, many hybrid algorithms are based on Boyer Moore algorithm, Kunth-Moris-pratt algorithms, Horspool algorithm, Wu-Manber algorithms and many more are also the alternatives of Boyer-Moore and KMP algorithms. Hybrids Algorithms are also designed for applications such as invasion detection, biological sequence analysis, imprecise string, virus scanning. The core purpose behind the pattern matching/string matching algorithms is to lessen number of comparisons of characters of text and to reduce the time required mainly for worst case and average case. Number of hybrid algorithms has been proposed to improve the said problem and improvise it.

We have divided this paper into two parts which are as follows; section 2 includes the working principles of fast hybrid string matching algorithms along with graphical representation of performance comparison with their parent algorithms and finally conclude with section 3.

II. METHODS AND MATERIAL

The data mining process is to be consisted of five steps.

- Problem statement and formulation Hypothesis
- Data gathering
- Data preprocessing
- Model estimation
- Model analysis

Purpose behind applying the hybrid approach in malware detection

- Speedy process for identification of malicious signatures and malware signatures.
- Implementation of hybrid approach thru data mining on large volume of data
- Protect user data
- To provide protection to system resources

- Provide application isolation

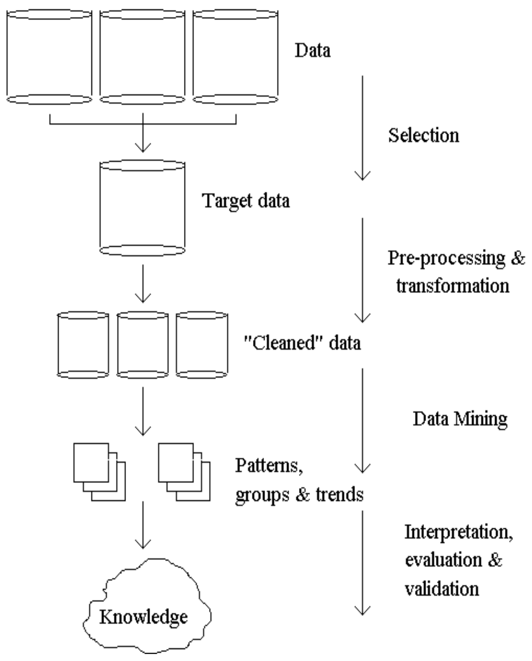

Data
Selection
Target data
Pre-processing & transformation
"Cleaned" data
Data Mining
Patterns, groups & trends
Interpretation, evaluation & validation
Knowledge

Figure 1: Data Mining Process

Features

- Strong security at the OS level through the Linux kernel
- Compulsory application storage state for all applications
- Secure inter-process communication
- Application signing
- Access with user granted permission at application level

Algorithm adopted for faster searching of patterns from database

Core idea behind the Boyer-Moore string-matching algorithm [BM77] is as follows. Let us assume that the pattern is of length is m. We start by matching it to the last character of the pattern against m, the m'th character of the text. If there is a mismatch between characters (and in most texts the probability of finding a mismatch is much greater than the probability of a match found), then we determine the rightmost occurrence of tm in the pattern and shift accordingly.

A. Theoretical Approach

When we come to the conclusion that the definitions checked is the malicious definition, the next step should be to add it to the database. Those definitions from database will be taken care of not finding the same and will match it in reference with the database. So the process becomes easy to detect them by just matching it with the database.

In this paper, they present a different approach that also uses the ideas of Boyer and Moore. The algorithm is quite simple Typical searches in this algorithm is focused rather than its worst case behaviour. To make the algorithm more faster than the existing one, we had to take some crucial engineering decisions take it in practice.

An earlier version of this algorithm was part of the second version of a grep [WM92a, WM92b], although the algorithm has not been discussed in [WM92b] and only briefly in [WM92a]. The current version is used in glimpse [MW94]. The design of the algorithm concentrates on typical searches rather than on worst-case behavior. So we decided to follow a new way to find the matching patterns from the database and the exact matching string from database if possible.

B. Proposed Algorithm

WM (Wu-Manber) multiple patterns matching algorithm adopts the idea of the hash technology and high competence categorizing ratio, which has the advantage of easy processing, logic clarity and operating efficiency. However, the existing WM algorithm-based methods for a achieving pattern matching for monitoring content security are all aim of English language texts

C. Overview of WM Algorithm

WM multi pattern matching algorithm applies hash method by pre-processing of model train set, creating three tables, SHIFT, HASH, and PREFIX. SHIFT is used to decide the shifting distance when mismatch characters or numbers found while matching it. HASH table and PREFIX table are used to choose which specific pattern needs to be matched when the SHIFT Table has a successful match. SHIFT table: Considering the size of the character block B, rather than simply a character, block transfer characters are used. In general the value of is assumed as 2 or 3, SHIFT build an index for all the possible characters the length of which is B. So the size of SHIFT table/block is the possible permutations of characters in B. using the value of SHIFT block the moving distance of a string of some certain B-characters is decided in the text, they will be the distance between the the characters at the right position of the certain B-characters and the tail of all patterns. Suppose X is B-length n character block of the current calculation and its hash value is i, we can think of two cases for this step: 1. X does not appear in any of the string pattern, string matching algorithm of the current text moves distance m-B +1 characters position, so we store m-B +1 in SHIFT [i].

Step 2. X string appears in some modes, in this case, the algorithm matches the rightmost position X that appears in the pattern string. Assuming X in P [j] appears in the position q, the position of X in the other modes at the same X position of the string is not larger than q. Thus, we should store m-q in SHIFT [i]. This is expressed by the formula:

Step 3. Finally, we will get SHIFT table. SHIFT table having new values displays the utmost safe distance of a long string of B when we are able to transfer the text. After ensuring the point pos and obtain the hash value i of the block B, when the SHIFT [i] > 0, pos = pos + SHIFT

D. Modified WM

Pattern matching algorithm is one of the Core algorithms in the recognition of the outside invasion engine for invasion prevention system. Efficiency of the intrusion avoidance system is determined by pattern matching algorithm. We have displayed the flow of the algorithm in the following Flow Chart.

Algorithm:

MWM has three stages, the preprocessing and the pattern search and storage.

(1) Before the MWM algorithm can be used, its hash and shift tables must be generated using a finite set of patterns provided beforehand. This is a one-time step that needs to be redone only when there are new sets of patterns. It should be pointed out here that the original WM algorithm adds a prefix table used to differentiate between patterns when their suffixes are the same but their prefixes are different. The first step in generating the tables is to find the minimum pattern length m from all signatures. This length m represents the number of characters that will be taken from the first letters of all signatures to form the shift table.

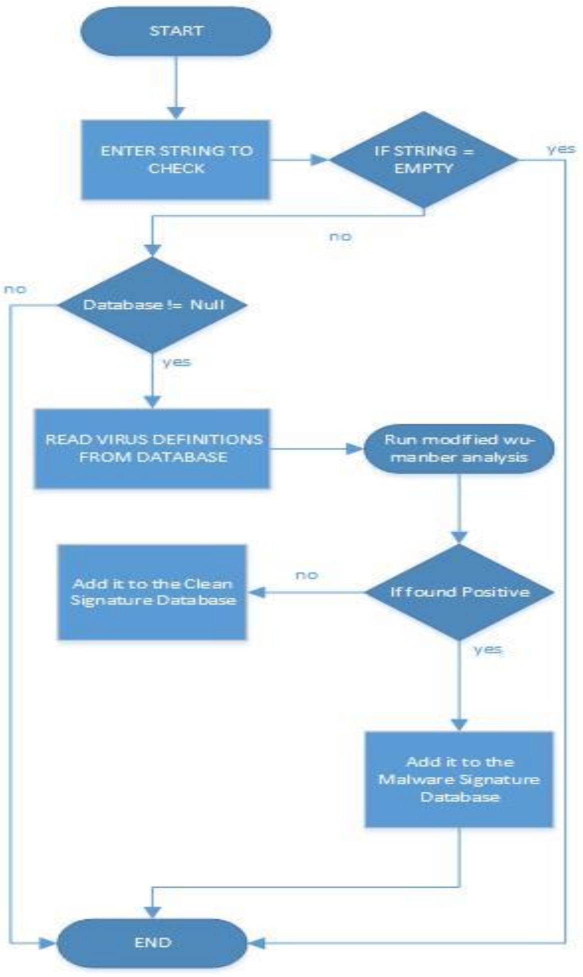

START
ENTER STRING TO CHECK
IF STRING = EMPTY
yes
no
Database != Null
no
yes
READ VIRUS DEFINITIONS FROM DATABASE
Run modified wu-manber analysis
Add it to the Clean Signature Database
no
If found Positive
yes
Add it to the Malware Signature Database
END

Figure 2: Flow Chart of Proposed Algorithm

(2) Divide Pattern into 2-2's blocks. Determine the max-size of prefix string as 3. Create HASH table "HASH" for each pattern set. Establish SHIFT table for pattern search where MWM can search different pattern sets concurrently. For every pattern sets, the search process is different.

(3) The shift table is built based on two variables: The first, B, is usually predetermined to a value of 2 or 3, albeit there are some other suggested methods for obtaining its value in accordance to the minimum length of the patterns and the number of rules provided. shift keys are assigned a shift value according to their location q in the pattern using the equation shift[key] = m – q.

(4) During the scanning process, a sliding window of size B scans through the searched string each time obtaining a sub-string of that size and getting its shift value from the shift table and shifting accordingly. Nonetheless, if the sub-string key does not exist in the shift table, a maximum safe

skip/shift of length (m – B + 1) is used. On the other hand, a shift value equal to zero requires access to the hash table and the traversal/search of all patterns that are linked to the key until a match is found or the list ends with no match. This iterative process is repeated until the whole string is completed.

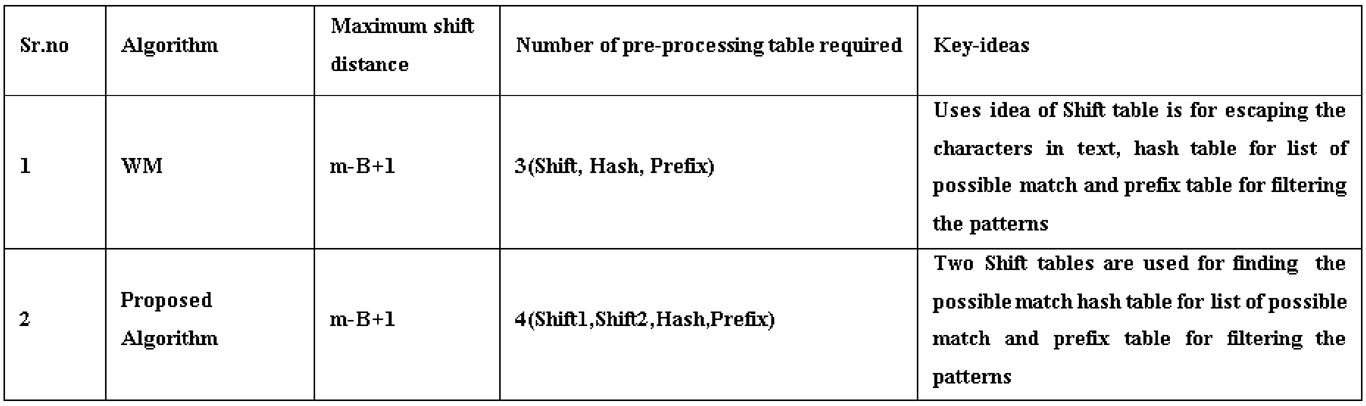
(5) Storage of last searched data will be stored into another hash table for future reference and that data will be searched first next time when the new strings to be searched are entered.

Advantages Proposed Algorithm Over Original WM:

1. To better the quality of the Boyer-Moore like Shift table, for each original pattern a pattern representative (hard to find substring with fixed length) is selected.
2. To increase the possibilities of shifting text sliding window, a second Boyer-Moore Shift table is computed.
3. To design a balanced Hash table for improving the scanning speed of possible matching patterns, a simple hash method with some random data property is added.

The advantages of changed algorithm over the previous explained algorithms is that the text scanning speed of the changed Wu-Manber algorithm is faster because with the use of second shift table exact string comparison is avoided most of the time.

Table-1 A Comparative Analysis of Basic Wu-Manber & Proposed Algorithm



Sr.no	Algorithm	Maximum shift distance	Number of pre-processing table required	Key-ideas
1	WM	m-B+1	3(Shift, Hash, Prefix)	Uses idea of Shift table is for escaping the characters in text, hash table for list of possible match and prefix table for filtering the patterns
2	Proposed Algorithm	m-B+1	4(Shift1,Shift2,Hash,Prefix)	Two Shift tables are used for finding the possible match hash table for list of possible match and prefix table for filtering the patterns

III. RESULTS AND DISCUSSION

Under the same test environment as described earlier (windows os, processor-core i3 and 4GB RAM), we compare the performance of WM and MWM.

Experiments are run on a set of virus signatures collected from different sites. The number of intrusions and the distribution of the intrusions across the traces vary from trace to trace. The set of traces are carefully selected to test the best and worst case performance of the algorithms. Table 1 lists the 5 traces used in the evaluation along with related statistics. The ugly traces “1” and “12” are chosen to test the worst case performance. They suffer from pathological performance due to their large sizes, the high feasibility of intrusion signatures in them, and the small skips which slow down the traversing window.

The bad traces “58” and “51” lie in the middle range with regards to the number of intrusion signatures and pathological performance. Finally, the good traces “bm”, “wm” and “mwm” are normal signatures that have not so high feasibility of intrusion signatures in them.

Table-2: Pattern Trace Analysis

Type of Trace	Serial Detections	Length(char)	Size	Instr-uctions (%)
Bad Trace				
20	458552	65623211	25.03MB	0.25
33	845220	61542202	24.44MB	1.25
Ugly Trace				
1	1244788	545114111	20.12MB	12.02
22	54785222	571245336	22.38MB	7.19
Good Trace				
BM	250	1052220	0.32MB	0.25
WM	2145	7542001	4.22MB	0.21
MWM	18599	3254479	1.25MB	1.45

The length of patterns is "6" characters all the time. Observe the effect on search time as the number of patterns increasing.

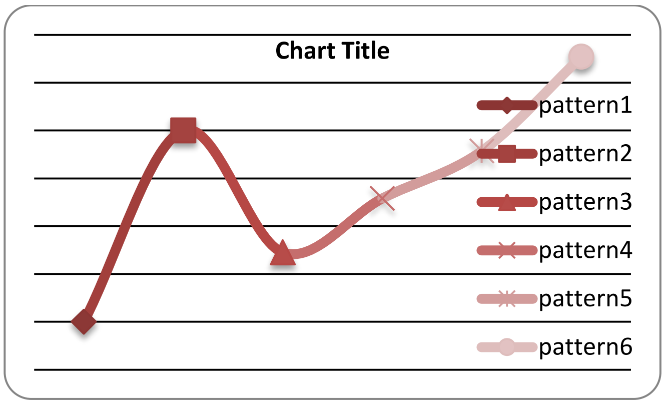


Figure 3. Comparison of String Patterns in Percentages

All three algorithms were thoroughly tested and found to provide excellent performance improvements having the execution times of a serial approach when using four threads. These performance improvements translate to swifter detections at better performance for pattern searching from a database.

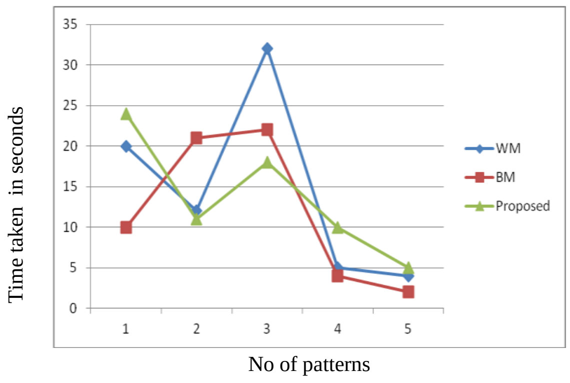


Figure 4. Comparision To Search A Pattern

IV. CONCLUSION

We purposed to find the fastest and efficient way to find a malware signature entered by user from the volume of malware signatures we have in database. The adopted research paper is suggesting wu-manber and Boyer-Moore algorithm for string matching. This paper proposed a method for backward hashing in a hybrid approach where we have proposed to find malicious signatures from a computer using data mining in an effective way. Because this approach is a mixture of multiple theories suggested by many of the programmer before, we will implement it to generate satisfactory results. After that the formal description of the program to obtain malicious signatures can be said appropriate for the purpose. We have adopted behaviour based detection method [BJL08].

V. REFERENCES

[1] [BJL08] Martin Boldt, Andreas Jacobsson, Niklas Lavesson, and Paul Davidsson. "Automate Spyware Detection Using End User License Agreements." isa, 0:445–452, 2008.

[2] [DM01] Data Mining: Concepts and Techniques. By : Jiawei Han and Micheline Kamber

[3] [ACK04a] Tony Abou-Assaleh, Nick Cercone, Vlado Keselj, and Ray Sweidan. "Detection of new malicious code using n-grams signatures." In Proceedings of Second Annual Con-ference on Privacy, Security and Trust, pp. 193–196, 2004.

[4] Computer Software and Applications Conference - Workshops and Fast Abstracts -(COMPSAC'04) - Volume 02, pp. 41–42, 2004.

[5] [Mal14]https://www.cert.gov.uk/wp-content/uploads/2014/08/An-introduction-to-malware.pdf

[6] [SD01] "Static Detection of Malicious Code in Executable Programs." Symposium on Re-quirements Engineering for Information Security (SREIS'01), 2001.

[7] [Bon93] Vesselin Bontchev. "Analysis and maintenance of a clean virus library." In Proceedings of the 3rd Internation Virus Bulletin Conference, pp. 77–89, 1993.

[8] [Ba89] Baeza-Yates R. A., ''Improved string searching,'' Software — Practice and Experience 19 (1989), pp. 257 271.

[9] [BM77] Boyer R. S., and J. S. Moore, ''A fast string searching algorithm,'' Communications of the ACM 20 (October 1977), pp. 762 772.

[10] [Kan02] Mehmed Kantardzic. Data Mining: Concepts, Models, Methods, and Algorithms.Wiley-IEEE Press, 2002.

[11] [WM92a] Wu S., and U. Manber, ''Agrep — A Fast Approximate Pattern-Matching Tool,'' Usenix Winter 1992 Technical Conference, San Francisco (January 1992), pp. 153 162.

[12] [WM92b] Wu S., and U. Manber, ''Fast Text Searching Allowing Errors,'' Communications of the ACM 35 (October 1992), pp. 83 91.

[13] [WM]http://wenku.baidu.com/view/096e6712a216147917112855.html

[14] [TA01] Danezis, George. "Traffic Analysis of the HTTP Protocol over TLS." Unpublished draft (2010).

[15] [TL01] Dierks, Tim. "The transport layer security (TLS) protocol version 1.2." (2008).

[16] [15] Dierks, Tim. "The transport layer security (TLS) protocol version 1.2." (2008).