CHARACTERISTICS BASED DETECTION OF INTERNET WORMS USING COMBINED MACHINE LEARNING METHODS AND WORM CONTAINMENT

CHAPTER 5

Detection and Containment of Self Propagating Monomorphic Characteristic worm based on Payload information using the Proposed DFF Method

5.1. Introduction

This chapter discusses the combined algorithm named as **D**eterministic Finite Automata with **F**uzzy Logic Classifier and **F**ilter-Ary Sketch (**DFF**) that is applied to detect the self propagating characteristics worms and containment of payload packets. Self propagating worms infect the network through packet payloads information exploiting the vulnerable applications. The payloads that exist in packets are detected using pattern matching method and classified into malicious and non-malicious payloads. Those malicious payloads are then blocked from further infection.

Each incoming packet payload is checked with the regular expression patterns. When the packet meets the regular expression pattern, the packet splits into sub packets. Each sub packet is constructed as states with the transition function using Deterministic Finite Automaton (DFA).The states are encoded and compressed to stateless and decoded to state using the Delayed Dictionary Compression (DDC) algorithm. DFA scans initial state with remaining states and detects the payload based on the redundancy of the states. Fuzzy logic classifier classifies the detected payloads and those payloads are hashed as malicious. Hashed malicious packets are measured using Jensen-Shannon divergence entropy and blocked using Filter-Ary Sketch. The proposed method contains various steps and is discussed in the next section.

5.2. Steps of the Proposed Contribution Two

The aim of the proposed contribution two is to achieve better detection of payload occurrence and containment of those packet payloads. The proposed method is mainly discussed in three steps.

Figure.5.1 shows the various steps of the proposed contribution two namely,

- Analysis and Preprocessing
- Detection and Classification of Internet Worms
- Containment of Internet Worms

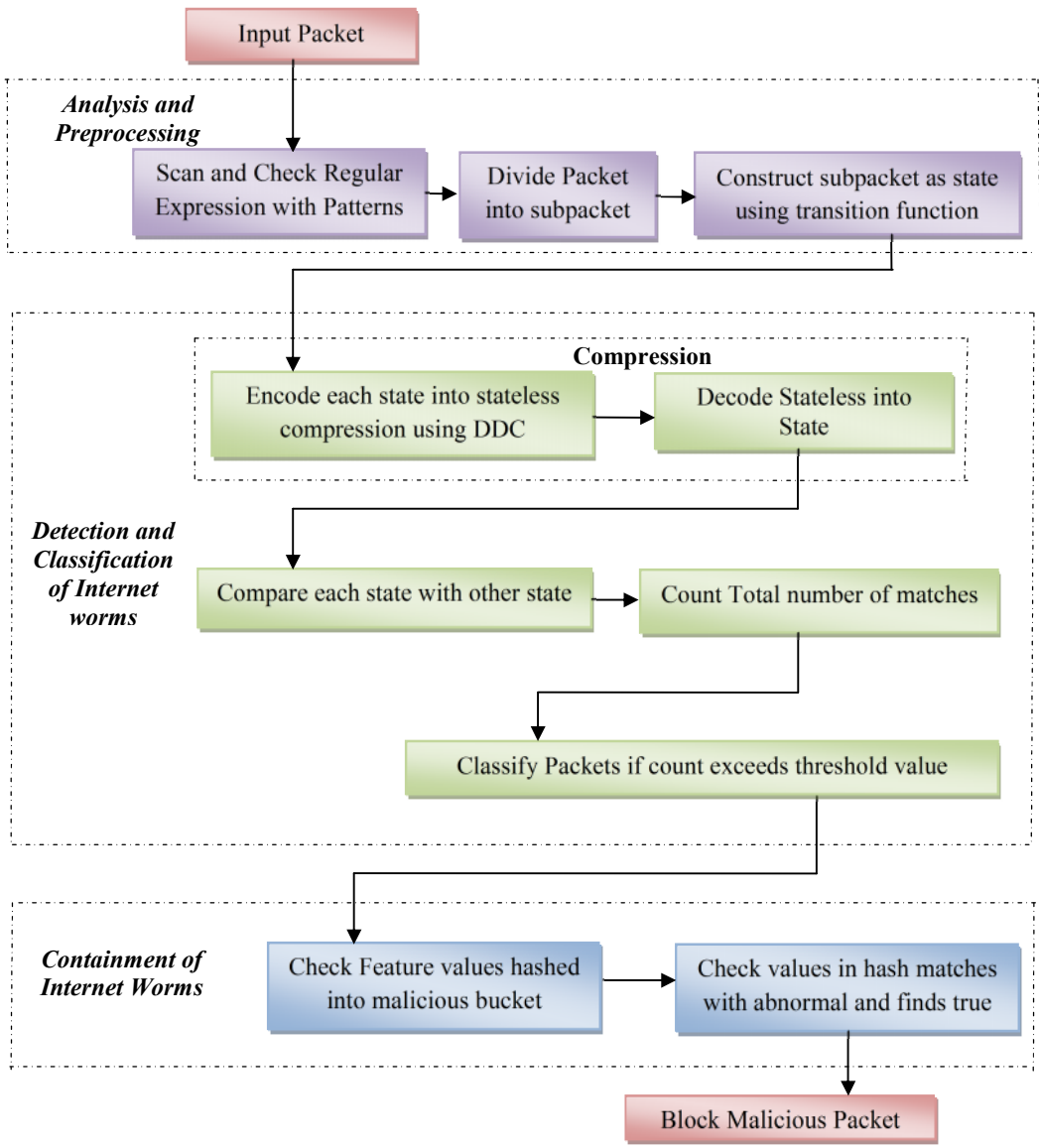


Figure.5.1. Block diagram of the Proposed Contribution Two

To detect and block the payload occurrence through vulnerable applications, a combined method called ***DFF*** method is proposed. DFF is a combination of ***D***eterministic Finite Automata, ***F***uzzy Logic Classifier and ***F***ilter-Ary Sketch method.

5.2.1. Analysis and Preprocessing

To analyze the repeated contents in the packet, Regular expression method is used. The packets are divided into sub packets and the repeated occurrence is analyzed. The regular expression and the patterns to split packets into sub packets are explained below.

5.2.1.1. Regular Expression

The sequentially arranged regular sets of characters that are structured for the search pattern are referred as Regular Expression. Each character in the string is categorized as either meta character or regular character with special meaning and literal meaning respectively in regular expression.

Regular expression of alphabet Σ are processed as

1) ε and every element of Σ is a RE
2) if e_1 and e_2 are of RE, then it results to $(e_1|e_2)$
3) if e_1 and e_2 are of RE, then it results to (e_1e_2)
4) if e is a RE then it results to e^*

The expressions which follow atleast any one of the rules above is a Regular Expression. The above listed are the syntax of Regular Expression. The clear note of a Regular Expression is the language denoted by RE that is defined using a function *l.* This is recursively defined as follows.

1) $l(\varepsilon)=\{\varepsilon\}$ and on every symbol, if *a* is the alphabet $l(a)=l\{a\}$
2) if e_1 and e_2 are RE, then $l((e_1|e_2))=l(e_1)\cup l(e_2)$
3) if e_1 and e_2 are of RE, then $l((e_1e_2))=l(e_1).l(e_2)$
4) if e is a RE, then $l(r^*)=l(r)^*$

By combining both the languages l_1 .l_2 the following equation is obtained

$$l_1 \cdot l_2 = \{w : w = ab \; for \; some \; a \varepsilon l_1 \; and \; b \varepsilon l_2\}$$

l_1^*, the Kleene closure of *l*, is the set of all string obtained by combining zero or more string from *l.* The equation of RE can be compressed by following the Kleene closure

operator " * " which has the largest precedence, combination and then " | " alteration. These both binary operators, ie., combination and alteration are left-associative.

Using these conversions, the RE *(a|(b(c*))d)* and equal, hence they match the same strings, and *a* or *a b* followed by sequence of zero or more *c*'s followed by *a d.*

Using the categorized set of strings textual material of a pattern can be identified. The sequence of pattern is expressed as a statement to represent the set of targets, in concise and flexibly to activate the text processing of general text files and specific textual forms automatically. Searching of regular expression patterns and identifying the matching text is pattern matching.

5.2.1.2. Patterns to split Regular Expressions

The collections of strings that are not listed in a specific format are defined as regular expressions. For scanning and analyzing packet payload with limited memory latency, the features listed in table.5.1 below are used.

Table.5.1. Patterns for Regular Expressions

Syntax	Meaning	Example
^	At the beginning, the pattern should be matched with this input.	^FH Shows FH starts with this pattern as input. F with this pattern '^', matches FH anywhere in the input.
|	OR relationship	F|H Represents F or H.
.	A single character wildcard	F. Represents end of the string.
?	Representing one or less quantifier	F? Represents F or an empty string.
*	Representing zero or more quantifier	F* Denotes an arbitrary number of Fs.
{}	Repeat	F{200} Means 200 Fs.
[]	A class of characters	[*ejk*] Represents a letter e*, j, or k.*
[^]	Anything but	[^\n] Represents any character. But not \n.

When the regular expressions meet the patterns listed in table.5.1, they are stored as sub packets containing limited number of strings. This overcomes the length restrictions in regular expression matching.

For payload detection, the regular expressions use DFA-based pattern matching approach. Compressing each state makes the DFA feasible and fits into the memory. Compression technique DDC is applied with DFA to overcome the memory limitation during the execution process.

5.2.2. Detection and Classification of Internet Worms

In this step, the proposed method Deterministic Finite Automata (DFA) with Delayed Dictionary Compression (DDC) scans every incoming packet in depth to detect the payload occurrence affecting the network. Regular expressions are analyzed and for matching regular expressions, DFA-based pattern matching is developed to detect payload. To achieve better matching speed, DDC is combined with DFA. DDC algorithm provides better increased speed links.

The detected payload packets are classified as malicious and non-malicious. The method applied for classification process is Fuzzy logic Classifier and is discussed below.

5.2.2.1. Matching Regular Expressions with DFA

The natural formalism used for regular expressions are finite automata and there are two types respectively, Deterministic Finite Automaton (DFA) and Nondeterministic Finite Automaton (NFA). In DFA, all transitions are deterministic, each transition leads to exactly one state. The analysis of regular expressions and developing memory-efficient DFA-based solutions providing high speed processing are discussed. Whereas in NFA, transitions are non-deterministic, each transition leads to subsets of states.

DFA is one of the finite automatons, in which all transitions are deterministic. DFA consists of a definite set of input symbols, denoted as $\sum$, definite set of states and a transition function, denoted as δ. $\sum$ consist of 2^8 symbols from extended ASCII code. Transition function δ gets the start state q_0 and an input symbol as an argument and enters the state. Each transition leads to exactly one active state. Regular expressions compare the packet with the pattern in the list. When it matches with the patterns, they split as states.

5.2.2.2. Deterministic Finite Automata (DFA)

Deterministic Finite Automata is a finite machine that allows or denies the finite set of string of symbols which produces a unique computation of the automata on every input string.

A deterministic finite automaton N is a 5-tuple $(S, \Sigma, \delta, q_0, E)$consisting of

- a finite set of states (S)
- a finite set of input symbols called as alphabet(Σ)
- a transition function$(\delta: S \times \Sigma \rightarrow S)$
- a start state $(q_0 \varepsilon S)$
- a set of accept states($E \subseteq S$)

Let $w=x_1,x_2,....x_n$ is a string of the alphabet Σ. The automaton N accepts the string w is the sequence of states, $r_0,r_1,r_2,....r_n$, exists in S with the following conditions:

i. $r_o=q_o$

ii. $r_{i+1}=\delta(r_i,x_{i+1})$, *for* $i=0,1,2,....n-1$

iii. $r_n \varepsilon E$

In the procedure,

- ***First step:***

  Machine starts with state q_o, and then each character of string w, the machine will transfer from one state to another state according to the transition function δ.

- ***Last step:***

  If the last input w makes the machine to stop in a particular accepting state

  Then the Machine accepts w

  Else the automaton denies the string

The string N, that is accepted by language l is denoted as $l(N)$.

5.2.2.3. Delayed-Dictionary Compression (DDC)

The Delayed Dictionary Compression algorithm generates the model M with a parameter additionally combined with Δ, that is, a non-negative integer. When there is a delay of Δ units, updating is done in dictionary either as characters or packets. From the input reading, dictionary D is a function of all $n - \Delta - 1$ units, for $n \leq \Delta - 1$. $\Delta = 0$ for every standard dictionary compression algorithm. The above defined approach is called Basic Delayed Dictionary Compression (BDDC).

The DDC algorithms are formed by the combination of BDDC and stateless compression. This algorithm produces better decoding latency using stateless compression. The encoder encodes the current characters T, encoding all characters till prior to last Δ characters. Each encoded packets point to a phrase. The encoded packet created as steps are stored in a dictionary as delay of Δ packets. DDC algorithm compresses the packets with the updated delay Δ proportional to network propagation delay. All encoded packets are compressed and stored in history except final Δ packets, as it precedes the currently encoded packet. Encoder transmits the entire encoded packet. Encoder transmits all the encoded step to the receiver. A Receiver receiving all packet headers specified in the history decodes the packets.

DDC is a general framework that can be applied for any dictionary algorithm; it consists of two main processes of encoding and decoding where the dictionary parser and the output parser are completely separated.

In this section, states obtained from the above process of DFA are given as input here. If there are large data used in DFA, the memory space allocated for it will be large in size. To reduce the memory and to decrease the computational time, DDC is used. DDC algorithm performs encoding, stateless compression and decoding to achieve decoding latency.

5.2.2.3.1. Encoder

The states matching with patterns and their transitions are given as input denoted by *I*. Then, compression algorithm maintains set of substrings called dictionary (D). The

parsing process for constructing the dictionary is called dictionary parser which is denoted as P_d; the obtained output parser is denoted as P_o.

The additional parameter which updates the dictionary with the delay is represented as Δ. It is a non-negative integer, constant, or adapted according to any rule that user chooses. For every standard DDC it is taken as zero.

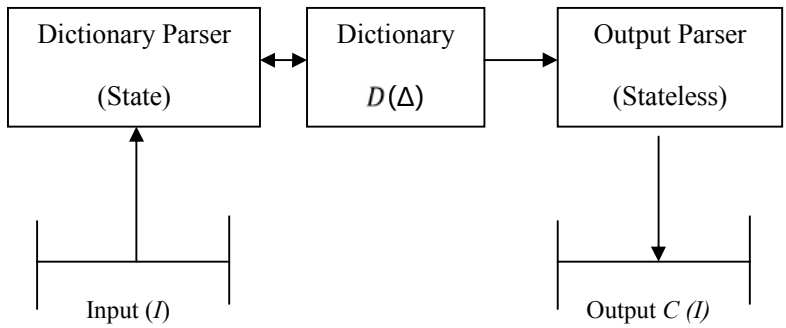


Figure.5.2. DDC- Encoder

From the above figure.5.2, the overall process of encoding is shown. Here the input *I* is given to the dictionary parser (P_d). Then dictionary $D(\Delta)$ consists of a set of substrings which is bi-directional. The packets accessed by P_d and *D* are given as a secret code to P_o. These secret codes are taken as compressed output.

5.2.2.3.2. Stateless Compression Algorithm

The Compression algorithm is applied to compress the packets independently during encoding. In stateless method, the compression and decompression are done independently for every packet. The receiver receiving the packets decompresses it regardless of its arrival order. This stateless compression minimizes the decoding latency.

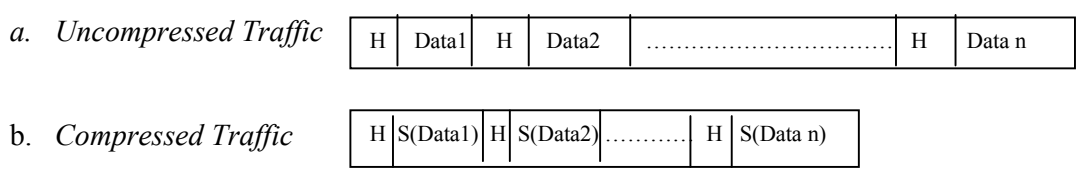


Figure.5.3. Stateless Compression Algorithm for packet payload

The traffic before compression is shown in figure.5.3.a; it consists of 'n' packets with each packet having its header and data. The traffic after compression is shown in figure.5.3.b; it consumes less memory compared to uncompressed. Each packet is compressed for less buffer size consumption.

5.2.2.3.3. Decoder

The secret codes obtained from the encoding technique of the DDC are given as input to the decoder. The decoding process is same as encoding where the reverse operation is done. The packet taken as input consists of secret codes. These codes are replaced with its corresponding phrases which build the dictionary.

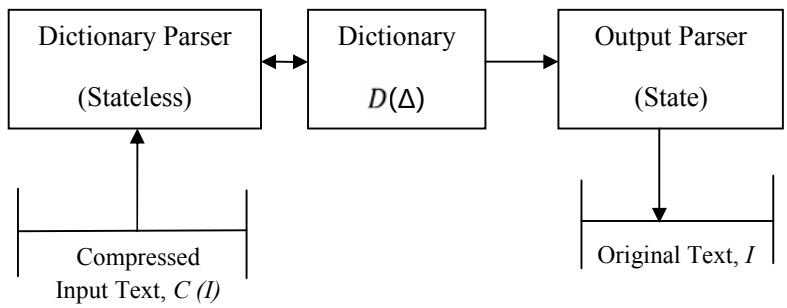


Figure.5.4. DDC- Decoder

From the above figure.5.4, the overall process of decoding is shown. Here the compressed text *C(I)* is given as input to the dictionary parser (P_d). Then dictionary $D(\Delta)$consists of a set of substrings which is bidirectional as in the encoder. The packets accessed by P_d and *D* of secret codes are replaced with its phrases in P_o. These phrases are the original text (*I*).

This compression technique makes the DFA fit in a reduced memory. This gives the way to match a large number of individual patterns with a lesser memory. The compressed states are monitored to detect payload.

5.2.2.4. DFA Matching to detect Payload

For the payload scanning, regular expressions and automata theory are directly applied. In packet payload scanning, input packets or substrings of input entering into the

network are matched with regular expression patterns. DFA faces complexity in recognizing all substring matches without any prior knowledge of start and end positions of substrings. In order to complete the matching process with DFA for all substrings, Exhaustive and Non-Overlapping matching styles are executed.

In Exhaustive Matching, pattern matches all the input substrings taken for matching and provides a set of results completely for the given input stream and the regular expression pattern. For example, for given pattern cb^* and input *cbbb*, the report will be three matches such as, *cb, cbb,* and *cbbb.*

For the matching process, let *M* be a function from a pattern P and a string *S* to a power set of S' such that,

$$M(P,S) = \{substring\ S'\ of\ S | S'\ is\ accepted\ by\ the\ DFA\ of\ P\}$$

Using this style of matching is expensive and matching every substring report is considered as unnecessary. To overcome the requirement of Exhaustive matching, Non-Overlapping matching is proposed.

In Non-Overlapping approach, for the matching process, let *M* be a function from a pattern *P* and a string *S* to a power set of S', such that,

$$M(P,S) = \{substring\ S_i\ of\ S | \forall\ S_i, S_j\ accepted\ by\ the\ DFA\ of\ P, S_i \cap S_j = \varphi\}$$

From the input strings, this matching process reports all non-overlapping substrings that match the pattern appearing in multiple locations of the input. For example, given pattern cb^* and input *cbbb*, the report provided by this match is only one and even the prefix 'cb' is overlapped thrice. Non-Overlapping matching for payload scanning provides better analyzing of pattern attacks found in the packet. This matching lacks in a memory-efficient DFA.

To handle pattern substring matching, One Pass Search execution model is created by DFA in this method. DFA created explicitly for extended patterns, matches the pattern anywhere with the input. Rather than scanning from beginning till end, DFA is able to begin its substring matching at different positions of the input. To suit the network applications, this one pass search approach achieves *O (1)* computation cost per character.

In this proposed contribution, for the packet payload scanning applications, DFA uses non-overlapping matches and one pass search. Figure.5.5 below illustrates DFA for regular expressions ^ab*cd?efi.gh

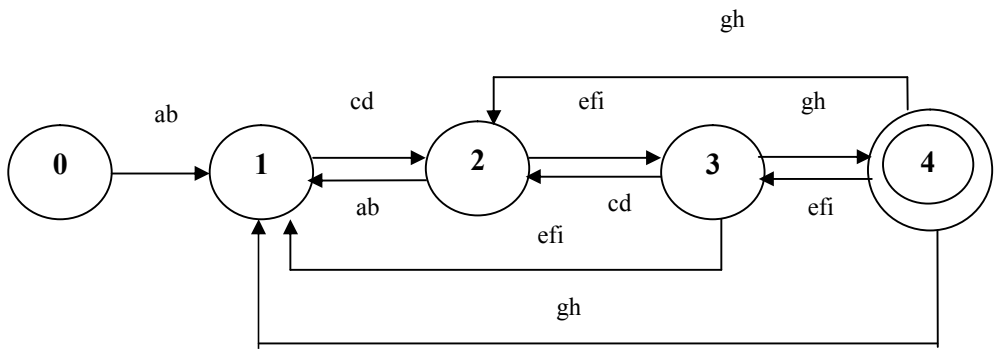


Figure.5.5. DFA illustrating Regular Expression

The detection step above clearly discussed the payload occurrence detection using DFA with DDC method. After the detection of payloads, they are classified as malicious payloads using Fuzzy Logic classifier and it is explained below.

5.2.2.5. Fuzzy Logic Classifier

Fuzzy logic is derived from fuzzy set theory dealing with reasoning which is approximate rather than precisely deduced from classical predicate logic. It is an extension of the classical Aristotelian – Boolean logic, means that it implements linguistic variables in a continuous range of truth-values, eliminating the problems of intermediate or uncertain values. The graphical representation of the concept "range of truths" is shown in figure.5.6.

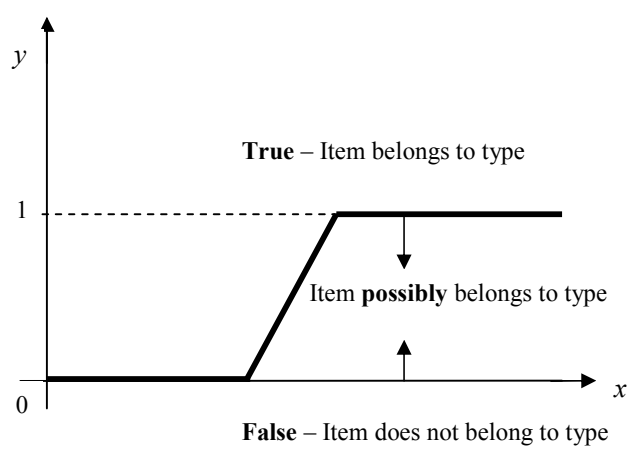


Figure.5.6. Graphical representation of the concept "range of truths"

Fuzzy logic is a form of many-valued logic that deals with inexact, relatively predetermined and precise reasoning. Compared to conventional binary logic, wherever variables may take on true or false values, the variables may have a truth value that ranges in degree from 0 to 1. Fuzzy logic is a comprehensive method to switch the concept of fractional accuracy, in which the accuracy value may range between absolutely true and absolutely false.

The notion central to fuzzy systems is that truth values (in fuzzy logic) or membership values (in fuzzy sets) are indicated by a value on the range [0.0, 1.0], with 0.0 representing absolute Falseness and 1.0 representing absolute Truth. A fuzzy system is distinguished by a set of linguistic statements based on expert knowledge. The professional acquaintance is habitually in the form of "*if-then*" conventions.

The classified malicious payloads are further blocked using the containment step and is discussed below.

5.2.3. Containment of Internet Worms

The classified malicious payloads are further blocked by the containment step that is to block the spread of payload occurrence into the network. The method applied to perform containment process is the Filter-Ary Sketch with Jensen Shannon Divergence (JSD) entropy to block the payload spread into the network.

5.2.3.1. Filter-Ary Sketch Method

The payloads classified by Fuzzy logic are applied in time window t, to store the identified payloads in rows. The row which has high value is labeled as reference payload. At the end of time t+1, to calculate the total number of occurrence of payload from the reference payloads, the entropy is used with JSD distribution.

Entropy is applied to detect the payloads and they are stored in malicious bucket, b_k using

$$H = -\sum_{i=0}^{n} \left(\frac{m_i}{m}\right) \log\left(\frac{m_i}{m}\right)$$

where m_i is the frequency of data item I and the data items of the stream in current window is referred by m, $m = \sum_{i=1}^{m} m_i$. Jensen Shannon Divergence (JSD) is applied to

compute the Meta-Data of payloads, which obtains the difference of a given distribution *p* to a reference distribution *Q* on the same space Ω by,

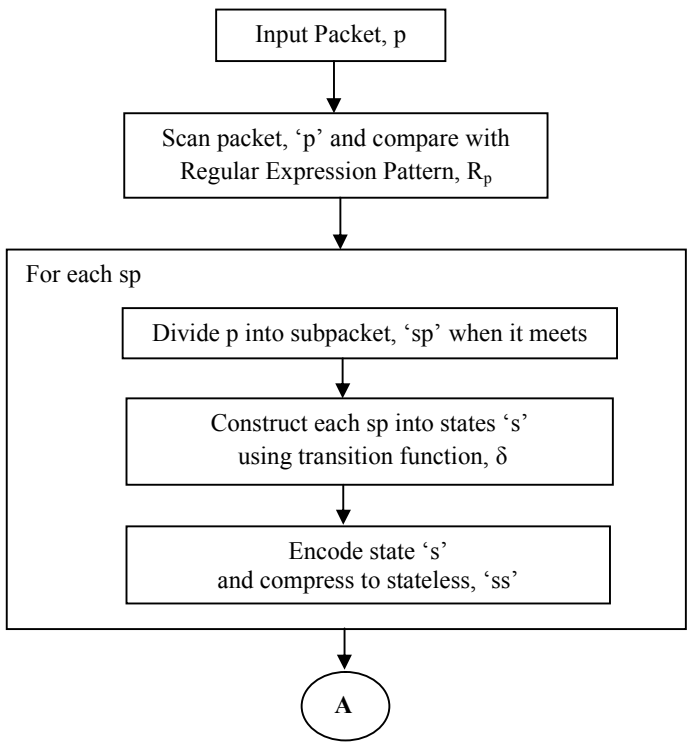
$$JSD(P \square Q) = \frac{1}{2}D(P \square M) + \frac{1}{2}D(Q \square M)$$

Using JSD distance, the maximum distances are identified and hashed as payloads in malicious buckets, $M_{i,k}$. The packets found in $M_{i,k}$ are blocked.

The different steps and the methods applied in proposed contribution two for detection and containment of self carried monomorphic characteristics worms are explained in detail above. The flow diagram of the proposed method is shown in the next section.

5.3. Flow diagram of the Proposed Contribution Two – DFF Method

The proposed method DFF detects the content payload occurrence using Deterministic Finite Automata with Delayed Dictionary Compression, classifies the malicious payloads using Fuzzy Logic Classifier and containment using Filter-Ary Sketch of JSD entropy method. The flow diagram of the proposed contribution two is shown below in figure 5.7.



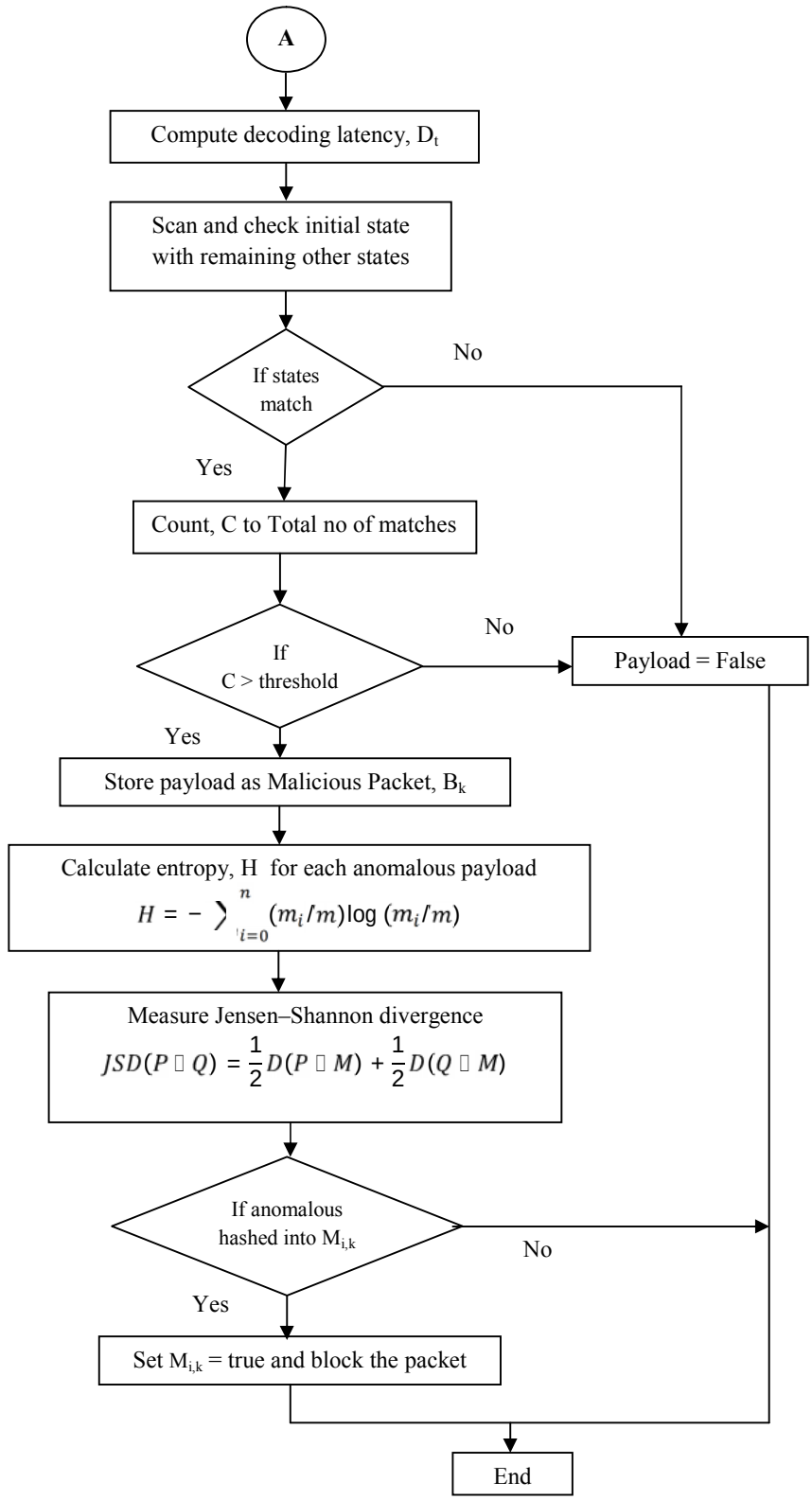


Figure.5.7. Flow diagram of the Proposed Contribution Two – DFF Method

The various steps involved to detect and contain that malicious payload content is discussed in the next section.

5.4. Steps involved in the Proposed DFF method

The various steps followed for the detection and containment of malicious payload content is discussed below:

Step 1: *Initialize the Regular Expression* R_e*, Scan and check each incoming packet P with* R_e*.*

Step 2: *Divide each packet P into subpacket* P_i*, when it meets the Regular expression pattern* R_p*.*

Step 3: *Construct the state transition function* δ*, gets the start state* q_0 *and an input symbol as an argument and enters the state.*

Step 4: *Apply encoding and decoding with the dictionary parser* $D(\Delta)$ *and compress each state to Stateless compressed packet with input* P_d *to obtain output parser* P_o*.*

Step 5: *Apply fuzzy logic to classify the payload occurrences.*

Step 6: *Compute the malicious payload into buckets and block those payloads applying JSD,*

$$JSD(P \square Q) = \frac{1}{2}D(P \square M) + \frac{1}{2}D(Q \square M)$$

The above following six steps are followed to detect the payload content occurrence infecting the network and containment of those detected payloads. The algorithm proposed is discussed in the next section.

5.5. Pseudo code of DFF Method

The algorithmic procedures applied for the detection of malicious content existing in the network and infecting during transfer is shown in the table.5.2.

Table.5.2. Pseudocode of DFF Method

```
Input: Packet p
Output: Block packet payload
Begin
    For each packet p transfer
    Scan and compare p with regular expression pattern R_p
    If (p matches R_p)
    Then Divide p into subpackets p
    end if
    for i=1 to n
       Construct sp(i) into state s(i) using transition function δ
       Compress stateless ss(i) = encode (s(i))
       s(i) = decode (ss(i))
      Compute decode latency
          for  i=1 to n
            for j=1 to n
                  Compare (s(i),s(j))
                  If (s(i) matches s(j))
                  Then Count=count + 1
                  end if
             until end of j
          until end of i
            If (count > threshold)
           Then Payload occurred
           end if
     For each dimension D
          If  decreases in JS divergence entropies of  D
         Then Normalized reduction is largest
              Declare D as anomalous dimension
         End if
      Next
     For each row i in the anomalous dimension of sketch r_{i,k}
        If j =  max_j| Given distribution _j-reference distribution _j| Store into Malicious packet MB_{i,k}
       End if
     Next
     Extract feature values for every packets in anomalous dimension
     For each row i in the anomalous dimension of the sketch
          If feature values is hashed into the MB_{i,k}
          Then R_{i,k}= true
          End if
                If all R_{i,k}=true
               Then Packet is blocked
               End if
       Until end of row
     Next
  Until end of packet
End
```

The section above briefly discussed the various steps involved in contribution two using the combined DFF method. The pictorial representation through flow diagram, steps involved and pseudo code of the proposed contribution two are also explained. The performance of the proposed DFF method is evaluated using the various parameters and the results obtained are explained in the section below.

5.6. Experimental Setup and Results

For experimentation, the applications like text and image files are downloaded from the Internet and are stored. The downloaded files are scanned using the Regular Expression Patterns. To detect the repeated occurrence, DFA method is applied. The detected files are classified as malicious packet payloads using Fuzzy Logic classifier. The malicious payloads are further blocked using JSD entropy. The evaluation is done using Java platform with the real data set.

.

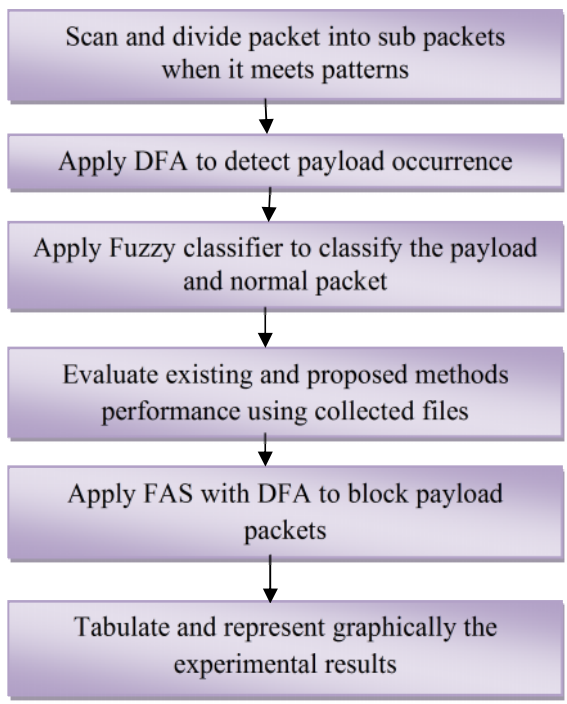


Figure.5.8. Experimental Methodology for Contribution Two

For validation of the proposed method, 500 files are used. The dataset contains 387 malware files and 113 normal files. The files are transferred between nodes and

repeated payload occurrences in the packet are detected and contained using proposed DFF method. The proposed contribution two DFF is implemented. The experimentation methodology is shown in above figure.5.8.

The proposed Deterministic Finite Automata with Delayed Dictionary compression and Fuzzy Logic Classifier (DDF) are compared with the existing General Frequent-common Gram Searching (GFGS) [56] method. The programs that are used for experimentation use two different file formats namely, text files and image files. A sample dataset is shown in Annexure II. The experiments are repeated for both single files and group of files containing five programs in each file for two different formats. The threshold is set to 3, specifying the total number of counts limit is three times only. The repetition of content is allowed and if it exceeds the threshold limit, the packet is said to be payload. The experiments are performed with different formats and the average of the values obtained in result are tabulated and shown in table.5.3 and figures 5.9 to 5.13.

Proposed **D**FA with **D**DC Algorithm and **F**uzzy classifier (**DDF**) method for detection is evaluated based on the parameters such as Memory Utilization, Time Consumption, Precision value, Recall value and Accuracy.

Table.5.3. Performance Comparison of Detection Results for Existing and Proposed DDF Method

Parameters	Existing GFGS (Tzu-Fang Sheu et al.)	Proposed DDF	% of Improvement
Memory Utilization (mb)	59622	**46371**	**22.22**
Time Consumption (ms)	22031	**3202**	**85.46**
Precision Value (%)	91.32	**97.95**	**6.76**
Recall Value (%)	77.89	**76.41**	**1.90**
Accuracy (%)	83.59	**83.39**	**0.23**

From the above table.5.3, it is observed that the proposed DFA with DDC Algorithm and Fuzzy classifier (DDF) provides better performance compared to that of the existing General Frequent-common Gram Searching (GFGS).

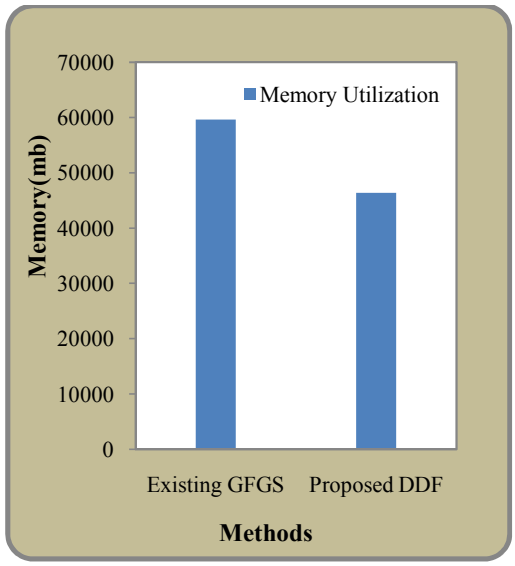


Figure.5.9. Comparison of Memory Utilization for Contribution Two

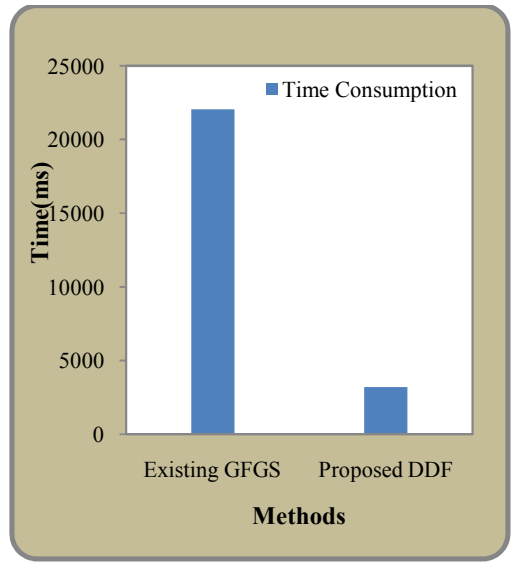


Figure.5.10. Comparison of Time Consumption for Contribution Two

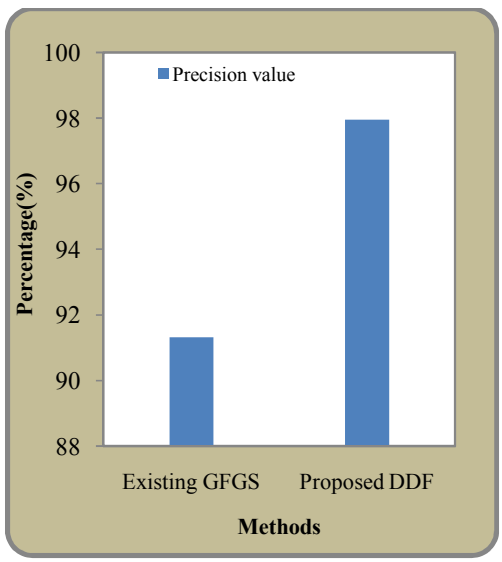


Figure.5.11. Comparison of results for Precision Value for Contribution Two

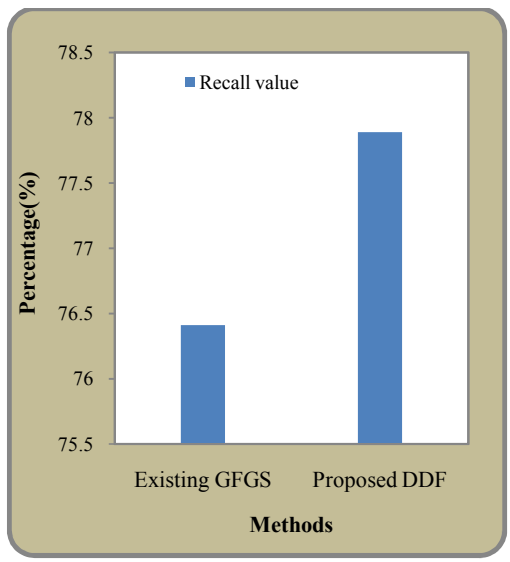


Figure.5.12. Comparison of results for Recall value for Contribution Two

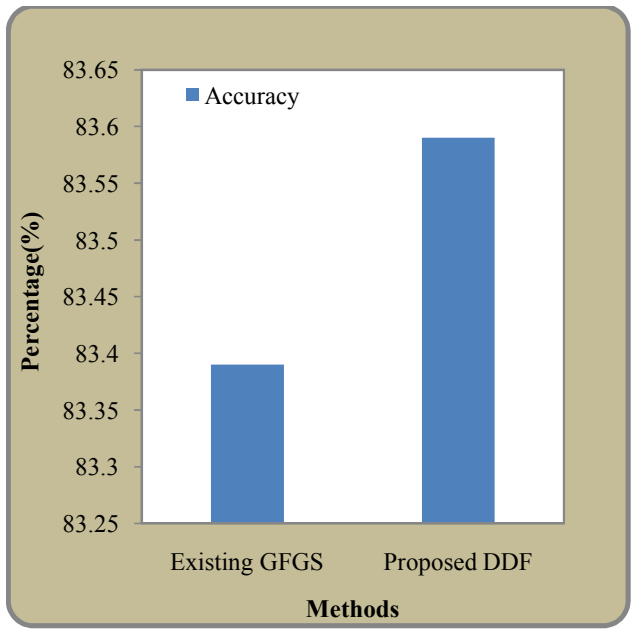


Figure.5.13. Comparison of results for Accuracy for Contribution Two

It is observed from figures.5.9 to 5.13, that the proposed method on the average has achieved minimum memory utilization and time consumption. The accuracy attained by the proposed method is also improved by 0.23%.

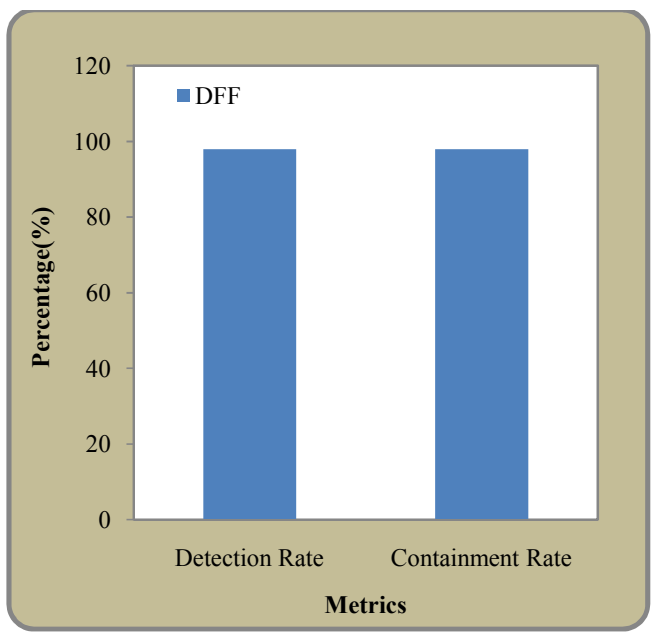


Figure.5.14. Results for Containment due to Contribution Two

The proposed Containment technique is evaluated using Detection Rate and Containment Rate and the result is shown in figure.5.14. It is very important to note that when the detection rate is 97.91%, the containment rate is 97.91%. The detected payload packets are blocked using the proposed DFF method. This shows that the containment rate is 100%, that is all the detected payloads are blocked.

The payload packets detected are completely blocked using the ***DFF*** Method over the time period of 1300 ms.

5.7. Chapter Summary

In networking applications, packet payload occurrence creates threats to the Internet users. In this chapter, self propagating worms exploiting through vulnerable applications like text and image files creating packet payload occurrences are detected and classified. The proposed contribution two provides better detection and containment of malicious payload occurrence existing in the packet through DFF method. DDC compression algorithm with DFA method is applied to achieve the compression overheads and it reduces the usage of memory. Fuzzy logic is used to classify the malicious packets detected through FAS method. The experimental results show the detection accuracy of 83.39% during detection and containment rate of 100% over 1300ms of time. In other words all the detected payloads are blocked in containment step. The detection accuracy is improved by 0.23% compared to the existing GFGS method. Though payload detection and containment provides better accuracy, they have some limitations like:

- Process of detection is slow, when large quantity of packet data is used
- When packet contents are encrypted, the analysis and detection lack

To detect at these circumstances, the network traffic behaviors are monitored and illegal propagation of traffic is detected to prevent the network from infection. The next chapter is about detection of second channel based propagating worms through botnet and its containment.