Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2014, Article ID 206867, 9 pages
http://dx.doi.org/10.1155/2014/206867

Research Article

Packet Payload Monitoring for Internet Worm Content Detection Using Deterministic Finite Automaton with Delayed Dictionary Compression

Divya Selvaraj and Padmavathi Ganapathi

Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore 641043, India

Correspondence should be addressed to Padmavathi Ganapathi; ganapathi.padmavathi@gmail.com

Received 26 June 2014; Revised 5 October 2014; Accepted 7 October 2014; Published 10 November 2014

Academic Editor: Tin-Yu Wu

Packet content scanning is one of the crucial threats to network security and network monitoring applications. In monitoring applications, payload of packets in a network is matched against the set of patterns in order to detect attacks like worms, viruses, and protocol definitions. During network transfer, incoming and outgoing packets are monitored in depth to inspect the packet payload. In this paper, the regular expressions that are basically string patterns are analyzed for packet payloads in detecting worms. Then the grouping scheme for regular expression matching is rewritten using Deterministic Finite Automaton (DFA). DFA achieves better processing speed during regular expression matching. DFA requires more memory space for each state. In order to reduce memory utilization, decompression technique is used. Delayed Dictionary Compression (DDC) is applied for achieving better speeds in the communication links. DDC achieves decoding latency during compression of payload packets in the network. Experimental results show that the proposed approach provides better time consumption and memory utilization during detection of Internet worm attacks.

1. Introduction

With rapid development, today the Internet has become more vulnerable to various threats and attacks such as intrusions, worms, viruses, spyware, and Trojans. Internet worm is a malicious code or program that exploits security holes and enters into the network without human interference [1, 2]. Internet worm is a self-propagating and fast spreading attack which has affected the Internet dramatically in the last few years. Moreover, malware attackers aim is to alter network traffic and create payload to cause infection at the host level [3]. Exploiting the software vulnerabilities, worms propagate and affect network services [4]. In order to protect the network from these attacks, effective defense mechanism is necessary.

The Morris worm in 1988 is the first network worm that infected DEC hosts and Sun3 operating systems to the large on Internet [5]. In 2004, Witty worm payload of 637 bytes is padded with data from system memory to fill the random size and a packet is sent out from source port 4000. After sending 20,000 packets, Witty seeks to a random point on the hard disk, writes 65 kbytes of data from the beginning of iss-pam1.dll to the disk. After closing the disk, the worm repeats this process until the machine is rebooted or until the worm permanently crashes the machine [6]. On October 2008, Conficker worm generated a large amount of network traffic and also caused user account lockouts [5]. Slammer and Code Red worms affected billion dollars and thousands of computers within an hour [3].

Network Intrusion Detection Systems (NIDS) have been adopted to defend against attacks that exploit the vulnerabilities of a protocol and attacks that seek to survey a site by scanning and probing. Presently, NIDS depends on the content based detection in minimizing the false alarm rate [7]. Scanning and probing attacks are detected by analyzing the network packet headers or monitoring the network traffic

connection attempts and session behavior. The Internet worm attacks create payload to a vulnerable service or application. These can be detected by inspecting the packet [8].

For earlier NIDS, string matching is essential because it contains a collection of strings represented as signatures. Different software and hardware solutions have been proposed for the string matching problems [9–12]. Various signatures based approaches are proposed for Internet worm detection [13–16]. Other than signatures, worms payload can be effectively detected by regular expressions through scanning every incoming packet. NIDS has a limitation for faster network links to match signatures with the incoming packets. To overcome this challenge, regular expression representation provides better network traffic monitoring by deep packet inspection [17]. Pattern matching methods provide better detection of Internet worms.

Deep packet inspection allows Network Intrusion Detection System (NIDS) to accurately identify the malicious payload occurring in the network during transfer of packets [17]. Pattern matching consumes more data volume, which takes more computation time and memory. Regular expression based DFA and NFA approaches can also be used for pattern matching in networks, but NFA has significant computation and storage complexities. NFA has multiple concurrent active states and bandwidth costs [18]. To overcome the challenge of existing approaches in scanning large number of packets with better speedup, DFA is implemented.

The proposed approach DFA for regular expression pattern matching inspects packet payload. DFA reduces the size of packets by splitting it into subpackets with the regular expression patterns to fit into the memory space. Accordingly, DFA applied with DDC increases speed in communication links and provides better decoding latency. Consequently, the proposed approach DFA with DDC provides better detection of the payload during transfer of packets in the network.

This paper is constructed as follows. Section 2 describes the previous approaches applied for Internet worm detection. Section 3 illustrates DFA used for regular expression pattern matching and the proposed approach in detail. In Section 4, the experimental results are shown comparing the proposed method with the existing approach. Section 5 concludes the paper.

2. Literature Review

Internet worms are causing a million-dollar damage by infecting thousands of machines within few minutes. Various defense mechanisms have been proposed by different authors to protect the network from Internet worm attacks. Some of the techniques proposed for detection of Internet worms are discussed below.

Wang et al. [19] implemented an approach for worm mitigation to validate the efficiency of the model through its extensive simulations. Considering the network-delay factor, the initial infection rate of active worms is detected. This approach also analyzes worm-free equilibrium point and derives basic reproduction number to quantify the guideline for effective worm defense. Amador and Artalejo [20] introduced an approach named block-structured state-dependent event (BSDE) to improve the computer network security. BSDE is used to find computer network infections and to boost up the computer security.

Khule et al. [21] proposed a novel method Netflow, to monitor traffic profiles in the network. The traffic statistics of packets received on an interface is counted as "flow" and stored in a dynamic flow cache. Toutonji et al. [1] proposed a mathematical model which combines both dynamic quarantine and passive benign worms for containment of worm propagation. To minimize the number of infected hosts by benign worms, the further research suggested by the author is to have quarantine measures.

Saikia et al. [7] implemented an approach to detect the worms using behavioral signature especially for improving network security. Yu et al. [22] found that C-worm propagation is detected using spectrum based detection scheme in distinguishing normal and abnormal background traffic. With the frequency domain, the pattern distinguishes C-worm traffic from normal traffic.

Yu et al. [23] introduced threshold-based, trace-back based, and spectrum based defense schemes. Threshold and trace-back are integrated to defend against static worms and the combination of above three schemes are used to defend dynamic self-disciplinary worms. The propagation patterns are analyzed for detection of worms. Zaki and Hamouda [24] proposed an anti-worm system to reduce effectively the spreading speed of infecting worms in network routers. WSRMAS (worm spreading reduction multiagent system) consists of a multiagent system to limit or even stop the worm spreading. Internet security is in real need for a realistic anti-worm system.

Table 1 summarizes various detection techniques proposed and the parameters used by proposing authors for their evaluation.

In Table 1, various techniques used for Internet worm attack detection are discussed with the proposed authors, parameters used, and its observed results. The observations indicate that the techniques proposed are efficient in increasing security for network, reducing worm propagation and identifying the attacks earlier.

Various signature based approaches have been developed by different authors for early worm detections. Cai et al. [25] proposed Wormsheild, a worm signature generation system to monitor the traffic. Distributed fingerprint filtering reduces aggregation traffic and the distributed aggregation trees improve load balancing to calculate fingerprint statistics. Wang et al. [26] analyzed network traffic based on patterns or signatures. The patterns at the network level are analyzed for detection of polymorphic worms to exploit the buffer overflow vulnerability. Based on both ploit-specific and vulnerability driven signatures, the zero day polymorphic worms are detected. Simkhada et al. [13] proposed a system to detect worms in a hierarchical manner by generating worm signatures automatically in large networks. Tang and Chen [14] proposed position-aware distribution signature (PADS) with expectation-maximization and Gibbs sampling algorithm for effective detection of worms. Tang et al. [16] proposed SRE signature based on exploitation of operating

Table 1: Review of literature for detection of Internet worms.

Year	Author	Technique(s) used	Parameters used	Observations
2009	Toutonji and Yoo [1]	Passive worm dynamic quarantine	λ_1 quarantine rate of infected hosts λ_2 and λ_3 quarantine rate of susceptible and passive hosts	Effectively decreases both the number of infectious hosts and worm propagation speeds. Widespread worm propagation maximizes quarantine rate.
2010	Yu et al. [23]	Game theory	(i) Infection rate (ii) False positive rate	Propagation patterns minimize the detection probability.
2010	Zaki and Hamouda [24]	Worm spreading reduction multiagent system (WSRMAS)	Infection percentage and Immunity percentage	Worm spreading stopped through multiagent.
2011	Yu et al. [22]	Spectrum-based scheme	(i) Detection rate (ii) Maximal infection ratio	Power spectral density shows low frequency bands.
2013	Amador and Artalejo [20]	Block-structured state-dependent event (BSDE)	(i) Scalar parameters K (population size) (ii) β (contact rate) (iii) γ (individual recovery rate) (iv) δ (external rate of infection) (v) ν (warning rate)	Reduces the propagation of virus by adding warning signals Better network security provided by BSDE model

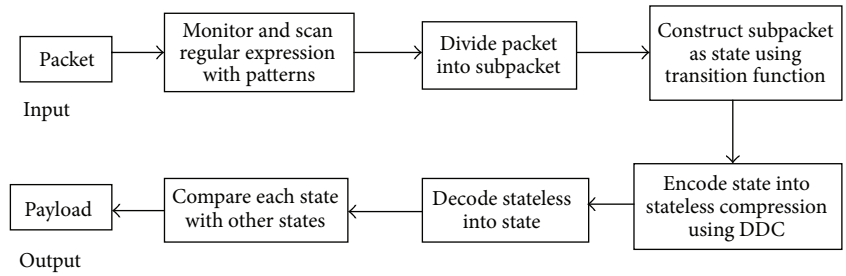


Figure 1: Proposed flow for pattern payload detection.

system and vulnerability of network services. Kong et al. [15] proposed semantics aware statistical (SAS) algorithm, to detect packets from the suspicious flow pool and generate worm signatures automatically. These signatures generated cannot survey for longer periods, instead regular expression is essential.

However, the above techniques have certain limitations in improving memory latency on detection of payload with regular expression. Thus the proposed approach is implemented by a compression algorithm to overcome the limitations of the existing approaches.

3. Proposed Methodology

The proposed approach DFA with DDC scans every incoming packet in depth to detect the payload occurrence of those 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 and minimizes decoding latency.

The steps followed for scanning and detecting the payload patterns during network transfer are given in Figure 1.

Figure 1 gives the proposed procedure. The techniques involved are regular expression matching, DFA, and DDC.

3.1. Matching Regular Expressions with DFA. The natural formalism used for regular expressions is finite automata and it is 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. While in NFA transitions are nondeterministic, each transition leads to subsets of states.

DFA is one of the finite automaton, in which all transitions are deterministic. DFA consists of a definite set of input symbols, denoted as $\sum$, and definite set of states and a transition function, denoted as δ. $\sum$ Consists 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.

3.1.1. Patterns to Split Regular Expressions. The collection 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 2 are used.

When the regular expressions meet the patterns listed in Pseudocode 1, they are stored as subpackets containing limited number of strings. This overcomes the length restrictions in regular expression matching. For payload detection, the regular expressions uses DFA-based pattern matching approach. To reduce the buffer size, compression technique is applied.

3.2. Proposed DFA with DDC. This section finds the solution to matching individual regular expression analyzed as DFA state in compressed stage. Compressing each state makes the DFA feasible and fits into the memory. Compression technique is applied to overcome the memory limitation by implementing DDC algorithm.

3.2.1. Delayed Dictionary Compression (DDC). The Delayed Dictionary Compression algorithm generates the model M with a parameter additionally combined with Δ, that is, a nonnegative 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 phases 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 phase 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

Table 2: Patterns for regular expression.

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.

```
Input: Packet p
Output: Payload
Begin
  For each packet p transfer
  Scan and compare p with regular expression pattern R_p
  If (p matches R_p)
    Divide p into subpacket sp
    for i = 1 to n
    {
        Construct sp(i) into state s(i) using transistion 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))
            {
               Count = count + 1
            }
         }
    }
        If (count > threshold)
            Payload occurred
  Until end of packet
End
```

Pseudocode 1: Pseudocode for proposed approach.

encoding and decoding where the dictionary parser and the output parser are completely separated. This gives to update the dictionary freely for parser process.

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.

(1) Encoder. The states matching with patterns and their transition 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 nonnegative integer, constant, or adapted according to any rule that user chooses. For every standard DDC it is taken as zero.

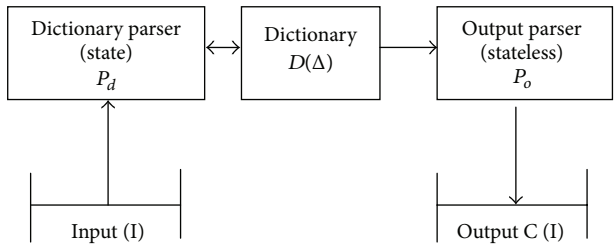
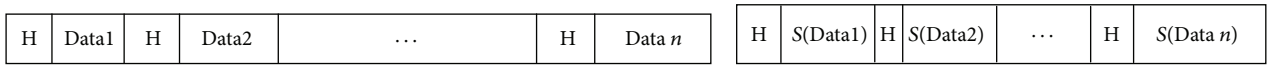


Figure 2: DDC-encoder.



(a) Uncompressed traffic (b) Compressed traffic

Figure 3: Stateless compression algorithm for packet payload.

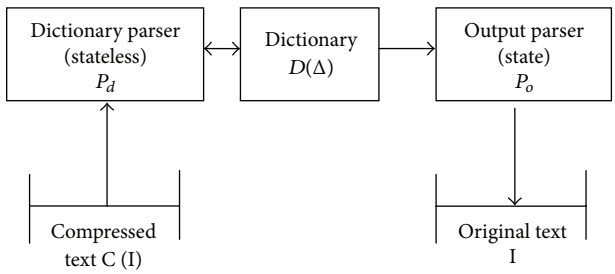


Figure 4: DDC-decoder.

From Figure 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 bidirectional. 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.

(2) Stateless Compression Algorithm. The Compression algorithm is applied to compress the packets independently during encoding. In stateless, 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.

The uncompressed traffic consists of "n" packets with each packet having its header and data. The compressed traffic in Figure 3(b) consumes less memory compared to uncompressed traffic. Each packet is compressed for less buffer size consumption.

(3) Decoder. The secret codes obtained from the encoding technique of the DDC are given as input. The decoding process is the same as encoding where the reverse operation of it is done. The packet taken as input consists of secret codes. These codes are replaced with its corresponding phrases which builds the dictionary.

From Figure 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.

3.2.2. 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 nonoverlapping 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 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

$$M(P,S) = \{\text{substring } S' \text{ of } S \mid S' \text{ is accepted by the DFA of } P\}. \quad (1)$$

Using this style of matching is expensive and matching every substring report is considered as unnecessary. To overcome the requirement of exhaustive matching, nonoverlapping matching is proposed.

In Nonoverlapping 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

$$M(P,S) = \{\text{substring } S_i \text{ of } S \mid \forall S_i, S_j \text{ accepted by the DFA of } P, S_i \cap S_j = \varphi\}. \quad (2)$$

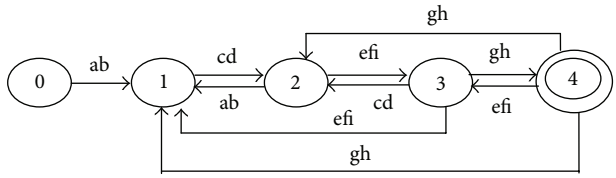


Figure 5: DFA illustrating regular expression.

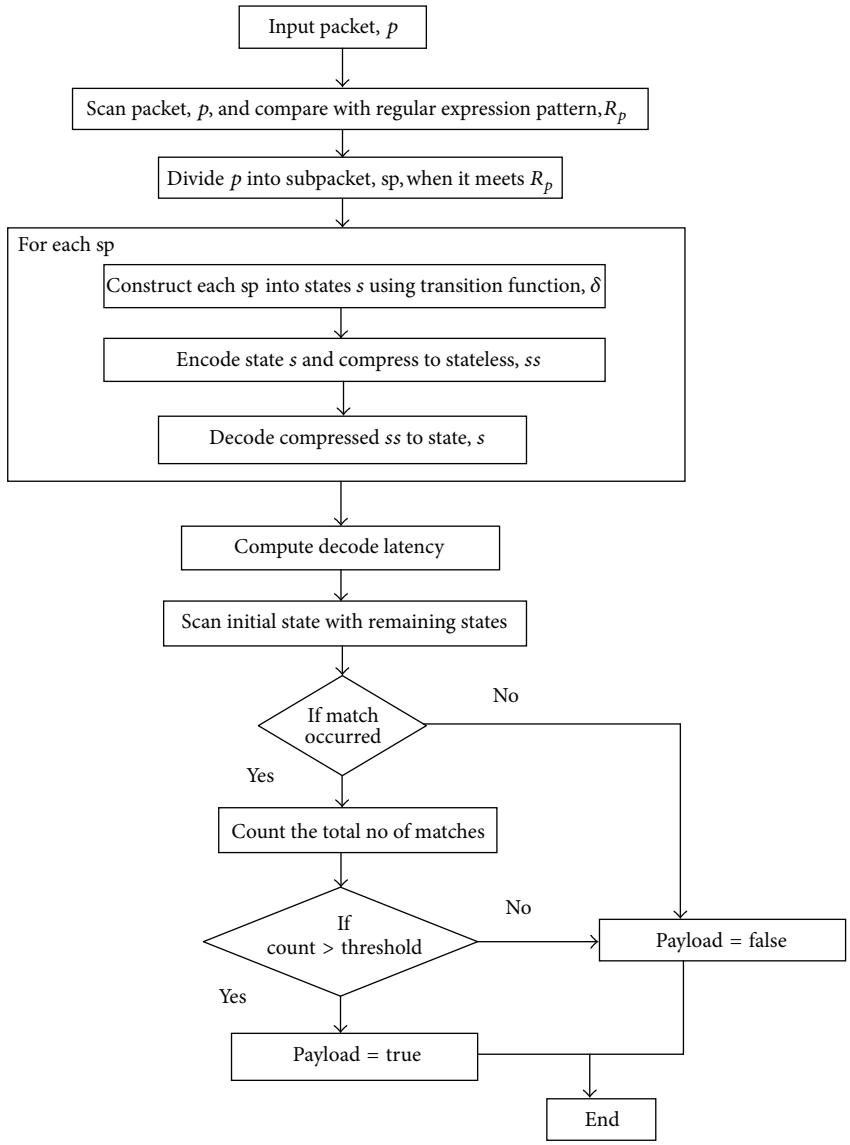


Figure 6: Flowchart for proposed approach.

From the input strings, this matching process reports all nonoverlapping 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*" overlapped thrice. Nonoverlapping matching for payload scanning provides better analyzing of pattern attacks found in the packet. This matching lacks in a memory-efficient DFAs.

To handle pattern substring matching, one pass search execution model is created by DFA in this paper. DFA created explicitly for extended patterns which 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 paper, for the packet payload scanning applications, DFA uses nonoverlapping matches and one pass search. Figure 5 illustrates DFA for regular expressions ^ab∗cd?efi.gh

Pseudocode 1 provides the pseudocode for the proposed approach. The approach integrates compression state with DFA technique for better memory latency.

In Pseudocode 1, the packets divided into states are converted to stateless. Then again the states are decompressed for achieving decode latency. Using DFA payloads are detected.

Figure 6 gives the flow diagram of the proposed approach DFA with DDC algorithm for monitoring and detecting the payloads created by Internet worms.

The proposed approach monitors and detects the packet payload created by Internet worms during the network transfer level to prevent its spread. Figure 6 shows the approach proposed that uses the DFA with DDC for monitoring and analyzing the packet payloads with the datasets trained. The DDC algorithm in the proposed approach detects the Internet worms in compressed state to overcome the memory space limitation.

4. Experimental Results

The approach proposed has been evaluated using the parameters like memory utilization and time computation. Memory consumed by the CPU during the detection of Internet worms based on payload is measured using memory utilization metric. Detection time is calculated to find the time consumed for detecting the Internet worms in the network using time utilization metric

$$\begin{aligned}\text{Memory Utilization} = {} & \text{Memory consumption of CPU at} \\ & \text{the end of the process} \\ & - \text{Memory consumption of CPU at} \\ & \text{the start of process} \\ \text{Time Consumption} = {} & \text{Finishing time of processing} \\ & - \text{Starting time of processing.}\end{aligned} \tag{3}$$

The evaluation is done using Java platform with the real data set. There are 500 sample data collected from Internet for monitoring and detecting the worm attacks. The dataset contains 387 malware files and 113 normal files. The data during data transfer are monitored and the attacks are detected through packet payload occurrence with the proposed DFA with DDC.

From Table 3, it is shown that the proposed DFA with DDC approach gives better results in terms of memory utilization and time consumption.

Figure 7 shows proposed pattern matching method in identifying packet payload with minimum memory requirements providing for resource scalability. Using DFA with DDC provides high speed matching and efficiency in memory utilization compared to the existing GFGS algorithm.

Figure 8 illustrates a comparison of computation time for the existing GFGS with EHAMA and proposed DFA with DDC. Figure 8 clearly shows that the proposed approach DFA with DDC algorithm gives lesser computation time than the existing GFGS with EHAMA.

Table 3: Proposed approach metric comparison.

Techniques	Existing GFGS with EHAMA	Proposed DFA with DDC	% of improvement
Memory utilization (KB)	59622	46371	22.22%
Time consumption (Sec)	22031	3202	85.46%

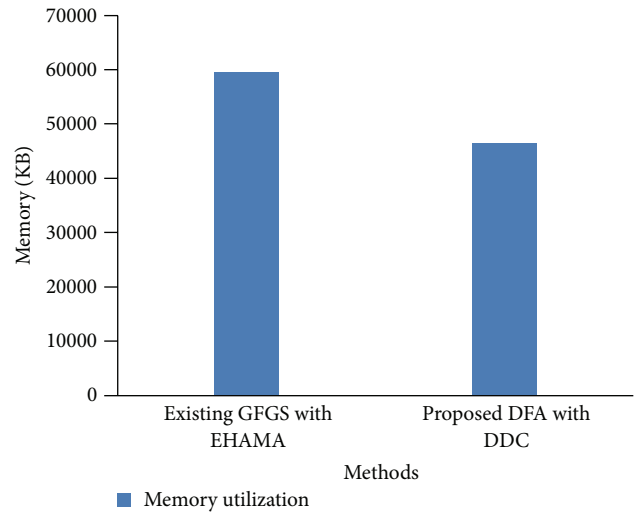


Figure 7: Comparison of memory utilization.

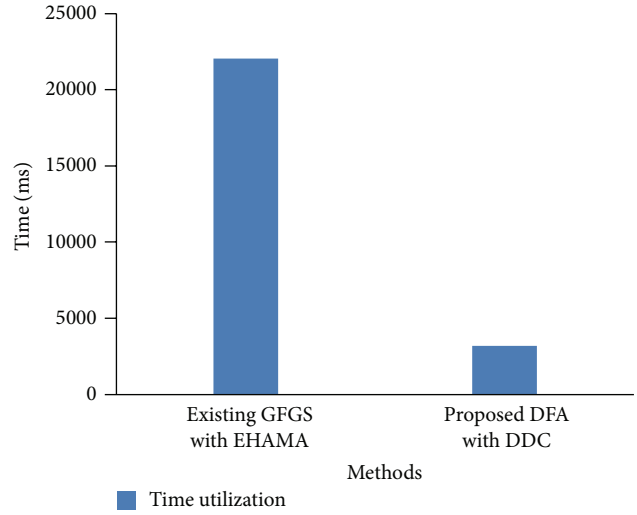


Figure 8: Time utilization comparison.

5. Conclusion

In networking applications, packet payload occurrence creates threats to the Internet users. In this paper, regular expression pattern matching with compression algorithm is implemented for monitoring packet payload created by Internet worms. DFA-based pattern matching implementation provides faster detection of payload occurrence. DFA focuses on detecting repeatable suspected packets and speeds up the scanning process. Additionally, DFA with DDC overcomes the compression overheads and reduces the usage of memory. DDC algorithm applied with DFA provides better decoding latency as well as speed in communication links. The experimental results in Section 4 shows that proposed method gives better memory utilization and time computation for detection of Internet worms compared to that of existing approach.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] O. Toutonji and S.-M. Yoo, "Passive benign worm propagation modeling with dynamic quarantine defense," *KSII Transactions on Internet and Information Systems*, vol. 3, no. 1, pp. 96–107, 2009.

[2] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 1, pp. 20–35, 2008.

[3] M. H. R. Khouzani, S. Sarkar, and E. Altman, "Maximum damage malware attack in mobile wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1347–1360, 2012.

[4] B. Bayoğlu and B. Soukpinar, "Graph based signature classes for detecting polymorphic worms via content analysis," *Computer Networks*, vol. 56, no. 2, pp. 832–844, 2012.

[5] O. A. Toutonji, S.-M. Yoo, and M. Park, "Stability analysis of VEISV propagation modeling for network worm attack," *Applied Mathematical Modelling*, vol. 36, no. 6, pp. 2751–2761, 2012.

[6] C. Shannon and D. Moore, "The spread of the Witty worm," *IEEE Security and Privacy*, vol. 2, no. 4, pp. 46–50, 2004.

[7] T. Saikia, F. A. Barbhuiya, and S. Nandi, "A behaviour based framework for worm detection," *Procedia Technology*, vol. 6, pp. 1011–1018, 2012.

[8] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *Recent Advances in Intrusion Detection*, vol. 3224 of *Lecture Notes in Computer Science*, pp. 203–222, Springer, Berlin, Germany, 2004.

[9] C. J. Coit, S. Staniford, and J. McAlerney, "Towards faster string matching for intrusion detection or exceeding the speed of Snort," in *Proceedings of the 2nd DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 367–373, June 2001.

[10] M. Fisk and G. Varghese, "Fast content-based packet handling for intrusion detection," UCSD Technical Report CS2001-0670, University of California, San Diego, Calif, USA, 2001.

[11] I. Sourdis and D. Pnevmatikatos, "Fast, large-scale string match for a 10Gbps FPGA-based network intrusion detection system," in *Field Programmable Logic and Application*, vol. 2778 of *Lecture Notes in Computer Science*, pp. 880–889, Springer, Berlin, Germany, 2003.

[12] I. Sourdis and D. Pnevmatikatos, "Pre-decoded CAMs for efficient and high-speed NIDS pattern matching," in *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '04)*, pp. 258–267, IEEE, April 2004.

[13] K. Simkhada, T. Taleb, Y. Waizumi, A. Jamalipour, N. Kato, and Y. Nemoto, "An efficient signature-based approach for automatic detection of internet worms over large-scale networks," in *Proceedings of the IEEE International Conference on Communications (ICC '06)*, pp. 2364–2369, July 2006.

[14] Y. Tang and S. Chen, "An automated signature-based approach against polymorphic internet worms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 879–892, 2007.

[15] D. Kong, Y.-C. Jhi, T. Gong, S. Zhu, P. Liu, and H. Xi, "SAS: semantics aware signature generation for polymorphic worm detection," *International Journal of Information Security*, vol. 10, no. 5, pp. 269–283, 2011.

[16] Y. Tang, B. Xiao, and X. Lu, "Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms," *Computers and Security*, vol. 28, no. 8, pp. 827–842, 2009.

[17] L. Yang, R. Karim, V. Ganapathy, and R. Smith, "Fast, memory-efficient regular expression matching with NFA-OBDDs," *Computer Networks*, vol. 55, no. 15, pp. 3376–3393, 2011.

[18] R. Smith, C. Estan, and S. Jha, "XFA: faster signature matching with extended automata," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 187–201, May 2008.

[19] F. Wang, Y. Zhang, C. Wang, J. Ma, and S. Moon, "Stability analysis of a SEIQV epidemic model for rapid spreading worms," *Computers and Security*, vol. 29, no. 4, pp. 410–418, 2010.

[20] J. Amador and J. R. Artalejo, "Modeling computer virus with the BSDE approach," *Computer Networks*, vol. 57, no. 1, pp. 302–316, 2013.

[21] M. Khule, M. Singh, and D. Kulhare, "Enhanced worms detection by Netflow," *International Journal of Engineering and Computer Science*, vol. 3, no. 3, pp. 5123–5127, 2014.

[22] W. Yu, X. Wang, P. Calyam, D. Xuan, and W. Zhao, "Modeling and detection of Camouflaging Worm," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 3, pp. 377–390, 2011.

[23] W. Yu, N. Zhang, X. Fu, and W. Zhao, "Self-disciplinary worms and countermeasures: modeling and analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 10, pp. 1501–1514, 2010.

[24] M. Zaki and A. A. Hamouda, "Design of a multi-agent system for worm spreading-reduction," *Journal of Intelligent Information Systems*, vol. 35, no. 1, pp. 123–155, 2010.

[25] M. Cai, K. Hwang, J. Pan, and C. Papadopoulos, "WormShield: Fast worm signature generation with distributed fingerprint aggregation," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 2, pp. 88–104, 2007.

[26] L. Wang, Z. Li, Y. Chen, Z. J. Fu, and X. Li, "Thwarting zero-day polymorphic worms with network-level length-based signature generation," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 53–66, 2010.