ISSN:0975-9646

Pattern Matching using Layered STRIFA for Intrusion Detection

P Prudhvi, H Venkateswara reddy

Department of Computer Science & Engineering, Vardhaman College of Engineering, Hyderabad, Andhra Pradesh, INDIA

***Abstract*-- With the advent and explosive growth of the global Internet adaptive/automatic network intrusion and anomaly detection in wide area data networks is fast gaining critical research and practical importance. In order to detect intrusions in a network, need efficient IDS. Deep packet inspection (DPI) has the ability to inspect both packet headers and payloads to identify the attack signatures in order to protect Internet systems. Regular expression matching, despite its flexibility and efficiency in attack detection, brings high computation and storage complexities to NIDSs, making packet processing a bottleneck. Stride finite automata (StriFA), a new family of finite automata, to accelerate both string matching and regular expression matching with reduced memory consumption. To increase the efficiency of StriFA, a layered approach of attack detection by using KDD 99 DARPA dataset is integrated with StriFA. We demonstrate that attack detection accuracy can be achieved by using StriFA and high efficiency by implementing the Layered Approach.**

Keywords-StriFA, Probe, DoS, R2L, U2R, NIDS

I. INTRODUCTION

Intrusions are the abnormal events happening in the computer system or network which attempts to compromise the confidentiality and availability of data or a system or a network. Intrusions are caused by attackers who seek to gain extra prerogatives by getting at a system from the internet; however they may be unauthorized user or the authorized users misusing their prerogatives. Intrusion detection is the mechanism of supervising events occurring in the networks to detect the abnormal behaviours of events i.e. intrusions. The most common approaches in intrusion detection system are anomaly detection and misuse detection. Anomaly detection can identify the activities that vary from the common behaviour, and thus have the potential to detect novel attacks.

An approach for detecting intrusions is to conceptualize both the normal and the known attack patterns for training a system, then performing classification of the test data. It integrates the advantages of both the signature-based and the anomaly-based detections, known as the *Hybrid System*. Hybrid systems are effective, subject to the categorization method used. They can be used to classify the unseen or new instances when occur, and then they assign one of the known classes to every test instance, because during training the system learns patterns and features from all the classes. But the only problem with the hybrid systems is the availability of labelled data. However, data requirement is also a concern for the signature, and the anomaly-based systems require anomalous and normal data, which are not easy to ensure.

A Network Intrusion Detection System (NIDS) scrutinize both packet headers and payloads to identify the intrusions in the networks in order to protect Internet systems. NIDS performs Deep Packet Inspection on network packets to identify attack signatures to secure the systems over the networks. Network security requires matching of huge volumes of data against large signature sets with thousands of strings in real time which can be done using pattern matching. Pattern matching is the core component, which works on the basis of string matching or regex matching. Also, this is the main bottleneck in NIDS for achieving the requirement of real-time processing at high speed. However, regular expression matching causes significantly high computation and storage complexities to NIDSs, hence its use in applications that require high processing speed and limited memory space is prohibited.

Pattern matching algorithms use finite state machines to identify among which most of them are derived from Deterministic Finite Automata (DFA). It can solve the pattern matching problem in time linearly proportional to the length of the input stream and independent of the number of strings in signature set. Deterministic finite automaton (DFA) and nondeterministic finite automaton (NFA) are two typical finite automata used to implement regular expression matching. In order to use DFAs practically in NIDS devices with limited memory resources, a large amount of work has been conducted on how to reduce memory consumption. Problem with DFA is high storage cost which is required to store transition rules.

To accelerate regular expression matching and enable deep packet inspection at line rate, Stride-based Matching, a novel acceleration scheme for regular expression matching, is proposed. StriFA [1] converts the original byte stream into a much shorter integer stream. Instead of matching the input stream byte by byte, StriFA method converts the input stream to integer stream for achieving higher throughput.

DARPA intrusion detection evaluation datasets have been employed to design and test intrusion detection systems in this paper. In 1999, recorded network traffic from the DARPA 98 Lincoln Lab dataset was summarized into network connections with 41-features per connection. This formed the KDD 99 intrusion detection benchmark in the International Knowledge Discovery and Data Mining Tools Competition. DARPA's KDD 99 dataset [2] provides the only publicly available labelled datasets for comparing IDS systems.

The DARPA's KDD 99 dataset is classified into 4 different attack groups. It is considered as the standard benchmark for intrusion detection evaluation. The training dataset of DARPA consist of approximately 5 million single association vectors, each of which contains 41 features. Empirical studies indicate that the feature diminution technique is capable of reducing the size of the dataset.

II. RELATED WORK

The research on intrusion detection and network security was going since 1980s. Many researchers proposed many schemes and frameworks to detect intrusions, which uses data mining techniques like association rules to recognize the patterns of the intrusions, and clustering methods, support vector machines etc.

Lee *et al.* [3][4][5] brought in data mining approaches for NIDS, which include association rules and regular sequences based on the classifiers by recognizing common patterns of program features and user behaviour. This method can recognize the features patterns, and define them as a packet and connection details. However mining of patterns should be limited to basic level as they require the number of records to be large; else they produce a large number of rules that makes the system more complex.

Clustering algorithms like k-means and fuzzy c-means in [6][7] are applied for attack detection, but problem with this is it works on calculating numeric distance between observations; therefore the observations must be numeric. And clustering methods are unable to identify the relationships between the features of a record, which further decrease the intrusion detection accuracy.

Ref [8] Support vector machines, maps the real valued feature non-linearly to higher dimensional feature space. They can be applied in both binary class and multi class classification, but generally aimed at distributed NIDS.

Literatures feat that throughput of multiple string matching can be improved using parallelism.

Dharmapurikar *et al.*[9] Presented a scheme with bloom filters mounted on-chip memory can examine multiple characters per cycle, achieved throughput up to gigabits per second with restrained memory usage. In worst case this method must access slow off-chip SRAMs regularly for accurate string equivalence.

Vespa *et al.* [10] propose a multiple stride matching method which is to group automata states/transitions into three coarse-grained and variable size blocks, blocks are identified based on DFA characteristics like prefix, linear trie and state dependencies, each block employ variable specific methods to optimize storage and matching speed performance. Reduced throughput in the worst case as this method is sensitive to rule set and input string.

Brodie *et el.* [11] enhanced the throughput of regex matching by expanding the alphabet set, ensuing an exponential increase in memory requirement in worst case. XFA uses appurtenant memory to reduce the DFA state explosion and attains great diminution rate. XFA is not suitable for real time applications on networks due to substantial start up overhead.

III. PROPOSED METHOD: LAYERED WITH STRIFA

We define four layers that represent to the four attack classes mentioned in the DARPA dataset. The Layered Approach is based on ensuring accessibility, privacy, and integrity of data and (or) services over a network. The main aim of using a layered approach is to decrease the computation and to increase the speed of intrusion detection. The time required to detect an intrusive transaction is substantial and this can be reduced by eliminating the overhead as the dataset is categorized into different layers. The overhead can be decreased by making the layers autonomous and adequate to block an intrusion without the help of a central decision-maker. Each layer is trained individually and then deployed consecutively. The four layers that are categorized into the four attack groups mentioned in the dataset are Probe layer, DoS layer, R2L layer, and U2R layer. Each rule set is generated by processing the respective trained dataset. The rule sets generated are the patterns for building Stride Finite Automata's, which process the input stream for intrusion detection. Feature selection [12] is also substantial for Layered Approach. The four attack groups and the features considered for their signs are:

A. Probe Attack

In this type of attacks an attacker scans a network of computers to gather information or to find known vulnerabilities. An attacker can use the information gained or through vulnerabilities with a map of machines and services that available on a network to look for exploits. So probe attacks are aimed at acquiring information about the target network from a source that is often external to the network. Hence, basic connection level features are considered such as the 'duration of connection' and 'source bytes' are significant. For probe attack, there are 5 significant features out of 41 features. The features are as follows

TABLE I
PROBE ATTACK FEATURES

Feature Number	Feature Name
1	Duration
2	protocal_type
3	Service
4	Flag
5	src_bytes

B. DOS Attack

In this type of attacks an attacker makes some computing or memory resources too busy or too full to handle authorized requests, or denies legitimate users access to a machine. For the DoS attack, traffic features such as the 'percentage of connections having the same destination host and same service' and packet level features such as the 'source bytes' and 'percentage of packets with errors' are considered. To detect DoS attacks, it may not be significant to know whether a user is 'logged in or not'. For DOS attack, there are 9 substantial features out of 41 features. The features are as follows

TABLE II
DOS ATTACK FEATURES

Feature Number	Feature Name
1	Duration
2	protocal_type
4	Flag
5	srs_bytes
23	Count
34	dst_host_same_srv_rate
38	dsthost_serror_rate
39	dst_host_srv_serror_rate
40	dst_host_rerror_rate

C. R2L Attack

In this type of attacks an attacker who does not have an account on a remote machine sends packets to that machine over a network and exploits some vulnerability to gain local access as a user of that machine by making the system violation. The R2L attacks are one of the most difficult attacks to detect, because both the network level and the host level features considered in order of detecting these attacks. So, both the network level features such as the 'duration of connection' and 'service requested' and the host level features such as the 'number of failed login attempts' among others are considered for detecting R2L attacks. There are 14 significant features out of 41 features for R2L attack. The features are as follows

TABLE III
R2L ATTACK FEATURES

Feture Number	Feature Name
1	Duration
2	protocal_type
3	Service
4	Flag
5	srs_bytes
10	Hot
11	num_failed_logins
12	logged_in
13	num_compromised
17	num_file_creations
18	num_shells
19	num_acess_files
21	is_host_login
22	is_guest_login

D. User to Root (U2R)

In this type of attacks an attacker starts out by accessing a normal user account on the system and is able to exploit vulnerability to gain root access to the system. The U2R attacks are difficult as they involve the semantic details that are very difficult to capture at an early stage. Most of the times U2R attacks are content based and they target an application. Hence, the aimed features for U2R attacks are 'number of file creations' and 'number of shell prompts invoked', while the features such as 'protocol' and 'source bytes' are ignored for U2R attack, there are 8 significant features out of 41 features. The features are as follows

TABLE IV
U2R ATTACK FEATURES

Feature Number	Feature Name
10	Hot
13	num_compromised
14	root_shell
16	num_root
17	num_file_creations
18	num_acess_files
19	is_host_login
21	is_host_login

After classifying the attacks into different layers we then perform intrusion detection by applying StriFA method. The StriFA method works as follows.

E. StriFA

Stride Finite Automata is a new family of automata which is similar to those of NFA/DFA but it can accept the integers as input characters. Its main idea is to convert the original byte stream into a much shorter integer stream and then match the integer stream with a variant of DFA, called StriDFA. To limit the size of the input stream and the number of comparisons we convert the input stream into a short integer stream which is called as the stride length stream. This can be done by selecting a frequently occurring character as a tag character and calculating the distance between these tag characters in the input stream. Now we feed this Sl stream to the StrideDFA for a potential pattern match. Once we found the potential match, then only there is a chance of complete string matching hence we go for neighbouring character match for the final match to confirm the identification of intrusion.

Tag: A character from the series of input streams or from training sets is taken as tag character which occurs regularly in them.

Stride Length: The distance between the two consecutive tagged characters is called stride length. The series of stride lengths generated from the input stream is called as a stride length stream.

1) StriFA for Multi String Matching: While processing, input stream is sent to the automation byte by byte, if FA reach any of its accepting states the match is found. The number of states visited is the length of the input stream on which time and memory access are determined which is bottleneck.

To increase the pattern matching speed and to reduce the memory accesses required, we need to reduce the number of states to be visited. To achieve this reduce the number of characters sent to FA.

Instead of comparing character by character, we pick a tag character from the input stream and feed the fingerprint of this tag characters to automation for processing. We use stride length of tag characters as fingerprints. The stride lengths extracted from rule set are equated with stride lengths extracted from the input stream for coarse grained match.

The snags with string matching are large alphabet set, regular expression support and false alarm rate.

2) Limiting Alphabet set by sliding window: To overcome the snag large alphabet set where a tag character is not found over a large distance, a fixed size sliding window is adopted. When there is no tag character within window distance, last character of the window is marked as fingerprint anchor (not a tag character but treated like one. The window size is send to StriFA as SL and the followed character is set as beginning of window and input send to strifa is limited to finite alphabet ∑={1,,,, w}.

3) Building StriFA: To enhance the regular expression support StriFA can be built by following steps

Compiling Regex to corresponding NFA

The regex rule set is compiled into equivalent NFA by conventional method where the NFA is converted to tag decisionFA instead of converting it into corresponding DFA.

Restructuring NFA to Tag decisionFA

Tag DecisionFA is directed graph consists of dotted transitions and solid transitions. In this, if transition input is tag character, then a solid transaction is drawn else to all those non tag characters dotted transitions are drawn. The resultant structure is called Tag DecisionFA.

Transforming Tag DecisionFa to StriNFA

In this a recursive algorithm is processed to build the transitions of StriNFA from the transitions of Tag DecisionFA. Starting from any state p in Tag decisionFA

Case 1: If a solid transition (pointing to state q) is reachable in l steps where l ≤ w, add a transition from p to q with label l (stride length).

Case 2: otherwise, if there is all-dotted-transition path of length w to state q, then add a transition from p to q with label w.

StriNFA to StriDFA

StriNFA is a special kind of NFA where input Alphabet is integers. The StriNFA to StriDFA conversion is similar to the NFA to DFA conversion using the structural induction algorithm.

4) Verification Module: Verification module is to decrease the false alarm rate; Since SL is a highly compressed stream from input stream there is a chance of false alarming, therefore to confirm the intrusion we have to perform the exact match. Unlike complete buffer match, only part of non tag characters send to the NFA considering the memory consumption. If the match found we conclude the intrusion, else the input stream recognized as normal Traffic.

F Layered Stride Finite Automata Architecture

The StriFA consists of three components: SL converters, StriFA matching engine and verification module. SL converters process the input stream and generate different SL streams according to the predetermined tags.

The StriFA engine is the core component, matches the input SL stream with the regex rule set similar to that of NFA/DFA.

The verification module is used to conclude the intrusion when there is a potential match in the StriFA engine found.

*1) Work flow:*The work flow of the entire scheme is the input stream is converted into Stride Length stream according the predetermined tags and then it is feed to StriFA matching engine. The rule set for each layer of attacks is given to the StriFA engine by which four virtual StriFA machines are built for each layer respectively. The Stride stream is matched with each StriFA built based on attack patterns respectively, the SL stream is processed by the next layer StriFA engine only when the potential match is not found in the previous layer StriFA engine, if there is a match found then it directly goes to verification module. Therefore the overhead of matching with other patterns is decreased. When there is no match found in all the StriFA machines or when the StriFA match is found and in verification module there is no match found, then it is considered as a normal traffic. The verification modules NFAs are also correspond to the attack classes. Hence each StriFA and NFA is built based on the respective layer attack patterns the memory consumption at each instance is less compared to those of regular attack detection mechanisms. Also event can be identified and classified as a particular attack type when an intrusion is matched with StriFA and verification module.

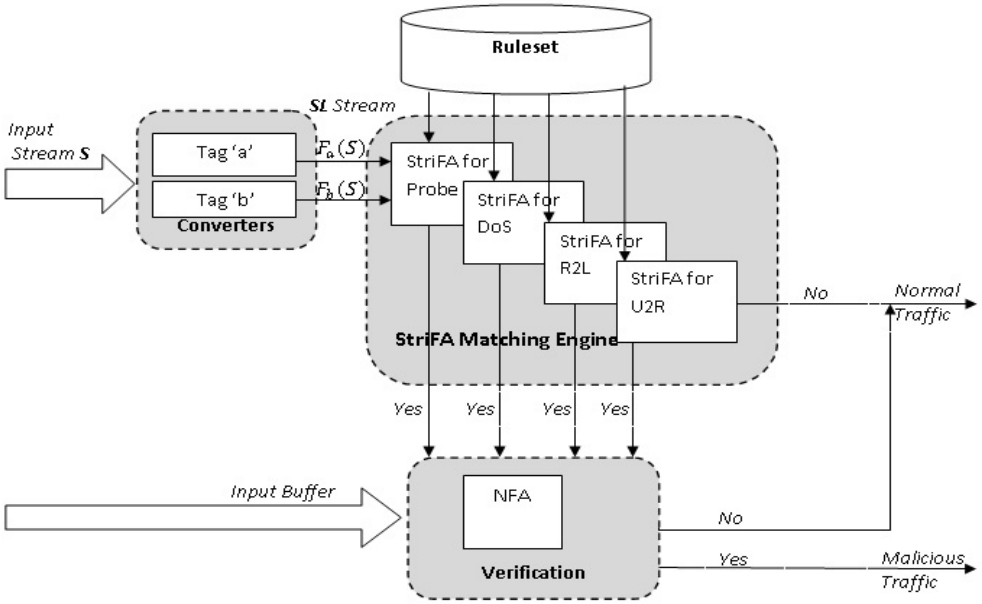

Figure 1: Layered StriFA Architecture

IV. CONCLUSION

In this paper, we presented a layered approach of Stride Finite Automata, improved mechanism in pattern matching for intrusion detection systems. The main principle of this method is to integrate the layered approach which classifies the attack types into four groups, thereby reducing the overhead of large rule set of patterns and reducing the memory consumption at instances. Therefore we can achieve high speed pattern matching or the packet processing with less cost of memory consumption.

REFERENCES

[1] Xiaofei Wang, Yang Xu, Junchen Jiang, Olga Ormond, Bin Liu, and Xiaojun Wang, "StriFA: Stride Finite Automata for High-Speed Regular Expression Matching in Network Intrusion Detection Systems" *Proc. IEEE SYSTEMS JOURNAL,* vol. 7, no. 3, Sept 2013

[2] DARPA Intrusion Detection Evaluation, https://www.ll.mit.edu/mission/ communications/cyber/CSTcorpora/ideval/docs/id99-eval-ll.html

[3] W. Lee and S. Stolfo, "Data Mining Approaches for Intrusion Detection," *Proc. Seventh USENIX Security Symp. (Security '98),* pp. 79-94, 1998.

[4] W. Lee, S. Stolfo, and K. Mok, "Mining Audit Data to Build Intrusion Detection Models," *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining (KDD '98),* pp. 66-72, 1998.

[5] W. Lee, S. Stolfo, and K. Mok, "A Data Mining Framework for Building Intrusion Detection Model," *Proc. IEEE Symp. Security and Privacy (SP '99),* pp. 120-132, 1999.

[6] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion Detection with Unlabeled Data Using Clustering," *Proc. ACM Workshop Data Mining Applied to Security (DMSA),* 2001.

[7] H. Shah, J. Undercoffer, and A. Joshi, "Fuzzy Clustering for Intrusion Detection," *Proc. 12th IEEE Int'l Conf. Fuzzy Systems (FUZZ-IEEE '03),* vol. 2, pp. 1274-1278, 2003.

[8] D.S. Kim and J.S. Park, "Network-Based Intrusion Detection with Support Vector Machines," *Proc. Information Networking, Networking Technologies for Enhanced Internet Services Int'l Conf. (ICOIN '03),* pp. 747-756, 2003.

[9] S. Dharmapurikar and J. W. Lockwood, "Fast and scalable pattern matching for network intrusion detection engines," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1781–1792, Oct. 2006.

[10] L. Vespa, N.Weng, and R. Ramaswamy, "Ms-dfa: Multiple-stride pattern matching for scalable deep packet inspection," *Comput. J.*, vol. 54, no. 2, p. 285, 2011.

[11] B. C. Brodie, D. E. Taylor, and R. K. Cytron, "A scalable architecture for high-throughput regular-expression pattern matching," in *Proc. ACM/IEEE ISCA*, vol. 34, no. 2, pp. 191–202, Jun. 2006.

[12] H. Güneş Kayacık, A. Nur Zincir-Heywood, Malcolm I. Heywood "Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets" *proc. IJCA JOURNAL,* vol. 31, no. 11, Nov 2011