IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 3, Ver. V (May-Jun. 2016), PP 01-06
www.iosrjournals.org

A Variation of Layered STRIFA Pattern Matching for Intrusion Detection

T.S. Urmila[1], Dr. R. Balasubramanian[2]
[1]*Department of Computer Science, Sourashtra College for Women, Madurai,*
[2]*Dean-PG, Karpaga Vinayaga Institue of Technology, Mathuranthagam*

Abstract: *Intrusion Detection System (IDS) is an effective security tool that helps to prevent unauthorized access to network resources by analyzing the network traffic and classifying the records as either normal or anomalous. Developing rules manually through incorporation of attack signatures are used in the detection of attacks. Finite State Automata (FSA) are used by many network processing applications to match complex sets of regular expressions in network packets. In order to make FSA-based matching possible even at the ever increasing speed of modern networks, multi-striding has been introduced. Stride finite automata (StriFA) a new family of finite automata, is 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 have converted symbolic named attributes with integer values in the dataset. This increases the accuracy rate and greatly reduces the error rate.*

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 through a series of events in the information system. Intrusions are caused by attackers who seek to gain extra privileges by getting at a system from the internet; however they may be unauthorized user or the authorized users misusing their rights. Intrusion detection is the mechanism of supervising events occurring in the networks to detect the abnormal behaviors of events i.e. intrusions. For example, a denial-of-service intrusion compromises the availability of an information system by flooding a constituent server with an overwhelming number of service requests to the server over a short period of time and thus denies or degrades the service to legitimate users. Another intrusion may compromise the integrity and confidentiality of an information system by gaining root privileges and then modifying and stealing information.

Existing intrusion detection techniques fall into two major categories: Misuse detection and Anomaly detection. The Misuse detection approach attempts to recognize attacks that follow intrusion patterns that have been recognized and reported by experts. Signature recognition techniques store the attack signatures i.e., on the detailed description of the sequence of actions performed by the attacker, perfectly match the observed behavior with these intrusion signatures and signal an intrusion when there is a match. In Misuse detection systems their effectiveness is strictly related to the extent to which Intrusion Detection Systems are updated with the signatures of the latest attacks developed and they are vulnerable to intruders who use new patterns of behavior or who mask their illegal behavior to deceive the detection system. This problem could be solved by designing general signatures that capture the "root-cause" of an attack, thus allowing for the detection of all the attack variants designed to exploit the same weakness. Unfortunately, general signatures designed by security experts usually generate high volumes of “false alarms” i.e., normal traffic events matching an attack signature.

Anomaly detection techniques establish a profile of the subject‟s (user, file, privileged program, host machine, computer network etc.)normal behavior (norm profile), compare the observed behavior of the subject with its normal profile, and signal an intrusion when the subject‟s observed behavior departs from its normal profile. Hence, anomaly detection techniques can detect both known and novel intrusions, if they demonstrate departures from a normal profile. For example, in a denial-of-service intrusion through flooding a server, the intensity of events to the server is much higher than the event intensity in a normal operation condition. In an intrusion through gaining root privileges, actions that an intruder takes to get into the information system and operation inside the information system are often different from actions of legitimate users in a normal operation condition. Hence, anomalies can be used to detect possible intrusions.

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 Intrusion Detection System passively observe the local network traffic and react to specific signatures (misuse detection) or statistical anomalies (anomaly detection). Examples of NIDS that employ misuse detection are Snort and Bro. One of the fundamental weaknesses of misuse-detection based NIDS is their inability to detect new types of intrusions. Anomaly detection techniques establish statistical profiles of network traffic and flag any traffic deviating from the profile as anomalous. But it needs complex structure with more knowledge.

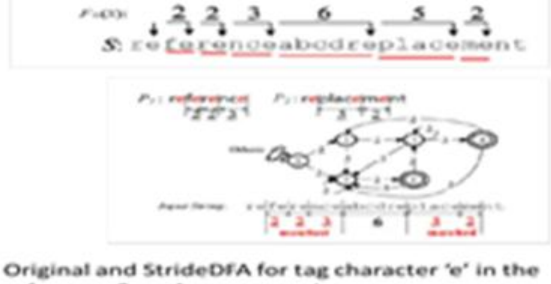
Network security requires matching of huge volumes of data against large signature sets with thousands of strings in real time which can be done using pattern matching. Pattern matching is the core component, which works on the basis of string matching or regex matching. Pattern matching algorithms use 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. DFA is fast and has deterministic matching performance, but suffers from the memory explosion problem. NFA, on the other hand, requires less memory, but suffers from slow and nondeterministic matching performance. Therefore, neither of them is suitable for implementing high speed regex matching in environments where the fast memory (e.g., cache or on-chip memory) is limited.

II. Strifa (Stride Finite Automata)

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 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. 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 (S1). Now we feed this Sl stream to the Stride DFA 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 neighboring character match for the final match to confirm the identification of intrusion.

StriFA for Multistring 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.

For example, if we select “e” as the tag and consider the input stream “referenceabcdreplacement,” then the corresponding stride length (SL) stream is Fe(S) = 2 2 3 6 5 2, where Fe(S) denotes the SL stream of the input stream S, “e” denotes the tag character in use. The underscore is used to indicate an SL, to distinguish it as not being a character. The SL of the input stream is fed into the StriDFA and compared with the SL streams extracted from the rule set. Clearly, the volume of processing to be performed by the DFA is reduced. The original DFA needs to process 24 input characters, while the new DFA only needs to process six input SLs. Of course, the DFA needs to be modified to handle the input stride (we call this new DFA variant StriDFA). The original DFA and StriDFA associated with the pattern P1 and P2 are given in Figure.1. The construction of the StriDFA in this example is very simple. We first need to convert the patterns to SL streams. The SL of patterns P1 and P2 are Fe(P1) = 2 2 3 and Fe(P2) = 5 2, respectively. After obtaining the SLs, we can then use the classical DFA construction algorithm to build the StriDFA. With increased speed, smaller memory consumption, and ease of implementation on existing platforms, the advantages of StriDFA are evident.



Original and StrideDFA for tag character 'e' in the reference & replacement string

III. Data Set Description

The DARPA‟s KDD99 dataset 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 and its features are grouped as,

1. Basic features - It encompasses all the attributes of TCP/IP connection and leads to delay in detection.

2. Traffic features - It is evaluated in accordance with window interval & two features as same host and same service.

(a) Same host feature - It examines the number of connections for the past 2 s that too from the same destination host. In other words, the probability of connections will be done in a specific time interval.

(b) Same service feature - It examines the number of connections in a particular time interval that too possess same service.

3. **Content features**

Dos & probe attack have frequent intrusion sequential patterns than the R2L & U2R. Because these two attacks include many connections to several hosts at a particular time period whereas R2L and U2R perform only a single connection. To detect these types of attacks, domain knowledge is important to access the data portion of the TCP packets. Ex. Failed login, etc. these features are called as content features.

List of KDD Cup dataset with Descriptions

No	Feature	Description
1	duration: continuous.	Duration of Connection
2	protocol_type: symbolic.	Connection Protocol (eg. TCP, UDP, ICMP)
3	service: symbolic.	Destination Service
4	flag: symbolic.	Status flag of the connection
5	src_bytes: continuous.	Bytes sent from source to destination
6	dst_bytes: continuous.	Bytes sent from destination to source
7	land: symbolic.	1 if connection is from/to the same host/port; 0 otherwise
8	wrong_fragment: continuous.	Number of wrong fragments
9	urgent: continuous.	Number of urgent packets
10	hot: continuous.	Number of "hot" indicators
11	num_failed_logins: continuous.	Number of failed logins
12	logged_in: symbolic.	1 if successfully logged in; 0 otherwise
13	num_compromised: continuous.	Number of "compromised" conditions
14	root_shell: continuous.	1 if root shell is obtained; 0 otherwise
15	su_attempted: continuous.	1 if "su root" command attempted; 0 otherwise
16	num_root: continuous.	Number of "root" services
17	num_file_creations: continuous.	Number of file creation operations
18	num_shells: continuous.	Number of shell prompts
19	num_access_files: continuous.	Number of operations on access control files
20	num_outbound_cmds: continuous.	Number of outbound commands in a ftp session
21	is_host_login: symbolic.	1 if login belongs to the "hot" list; 0 otherwise
22	is_guest_login: symbolic.	1 if login is the "guest" login; 0 otherwise
23	count: continuous.	Number of connections to the same host as th current connection in the past 2 seconds
24	srv_count: continuous.	Number of connections to the same service as th current connection in the past 2 seconds
25	serror_rate: continuous.	% of connections that have "SYN" errors
26	srv_serror_rate: continuous.	% of connections that have "SYN" errors
27	rerror_rate: continuous.	% of connections that have "REJ" errors
28	srv_rerror_rate: continuous.	% of connections that have "REJ" errors
29	same_srv_rate: continuous.	% of connections to the same service
30	diff_srv_rate: continuous.	% of connections to the different services
31	srv_diff_host_rate: continuous.	% of connections to different hosts
32	dst_host_count: continuous.	Count of connections having the same destination host
33	dst_host_srv_count: continuous.	Count of connections having the same destination and using the same service
34	dst_host_same_srv_rate: continuous.	% of connections having the same destination host and using the same service
35	dst_host_diff_srv_rate: continuous.	% of different services on the current host
36	dst_host_same_src_port_rate: continuous.	% of connections to the current host having the same src port
37	dst_host_srv_diff_host_rate: continuous.	% of connections to the same service coming from different hosts
38	dst_host_serror_rate: continuous.	% of connections to the current host that have an S0 error
39	dst_host_srv_serror_rate: continuous.	% of connections to the current host and specified service that have an S0 error
40	dst_host_rerror_rate: continuous.	% of connections to the current host that have as RST error
41	dst_host_srv_rerror_rate: continuous.	% of connections to the current host and specified service that have an RST error

KDD Cup 99 data set contains 22 attack types and their names and the related features for a particular attack are defined in the table below. For all 22 attacks, the related features are calculated by enabling the threshold value. If the attribute satisfies the specified constraints then the attribute is chosen as the related features of particular attack.

IV. Attacks Descriptions

Dos attack – It is a kind of attack where the attacker makes processing time of the resources and memory busy so as to avoid legitimate user from accessing those resources. 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 „percentages 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". **Smurf, teardrop, pod, back, land, Neptune** is classified as DOS attacks**.** For DOS attack, there are 9 substantial features out of 41 features.

U2R attack – Here the attacker sniffs the password or makes some kind of attack to access the particular host in a network as a legitimate user. They can even promote some vulnerability to gain the root access of the system. The U2R attacks are difficult as they involve the semantic details that are very difficult to capture at an early stage. **Buffer_overflow, load module, Perl, root kit** are classified as User to Root attacks**.** 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

R2L attack – Here the attacker sends a message to the host in a network over remote system and makes some vulnerability. 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 Remote to Local attacks. **ftp_write, guess_password, imap, spy, multihop, phf, warezclient, warezmaster** are classified as R2L attackd. There are 14 significant features out of 41 features for R2L attack.
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.

Probe attack – Attacker will scan the network to gather information and would make some violation in the future. 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. **ipsweep, portsweep, nmap, Satan** are classified as Probe attacks. For probe attack, there are 5 significant features out of 41 features. The features are as follows

DOS ATTACK FEATURES

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

U2R ATTACK FEATURES

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

R2L ATTACK FEATURES

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

PROBE ATTACK FEATURES

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

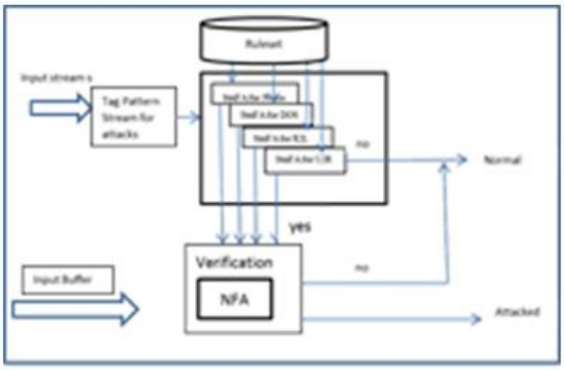
The Table below represents the rule structure for the KDD Cup 99 data set. Using this rule structure the data set can be easily classified in the future. If any new type of attack is found it can also be added in the in this profile for better classification results.

Rule no	Attack description	Attack type
1	[illegible]	Smurf
2	[illegible]	Neptune
3	[illegible]	Nmap
4	[illegible]	Back
5	[illegible]	Satan
6	[illegible]	teardrop
7	[illegible]	ipsweep
8	[illegible]	portsweep
9	[illegible]	ftp_write
10	[illegible]	guess_passwd
11	[illegible]	Imap
12	[illegible]	Multihop
13	[illegible]	Spy
14	[illegible]	warezclient
15	[illegible]	Buffer_overflow
16	[illegible]	warezmaster
17	[illegible]	load module
18	[illegible]	Perl
19	[illegible]	root kit
20	[illegible]	Land
21	[illegible]	Pod
22	[illegible]	Phf

V. Variation of Strifa

We have randomly selected 1000 data from the dataset. The preprocessing step involves the mapping of symbolic valued attributes into numeric valued attributes. Symbolic features like protocol type (3 different symbols), service (23 different symbols) and flag (7 different symbols) are mapped to integer values ranging from 0 to N-1 where N is the number of symbols. Then the KDD ‟99 dataset is fragmented into 4 subsets, each containing records of normal and a specific attack category.

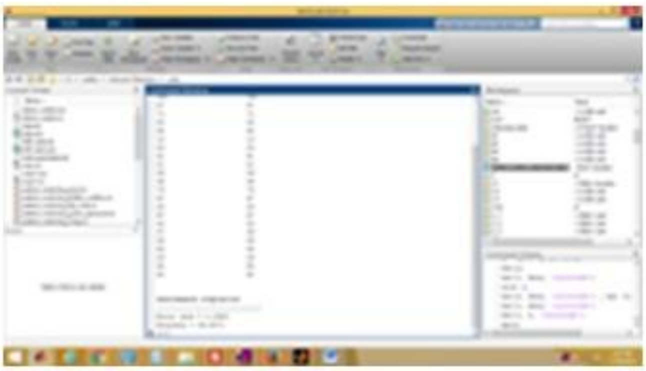
The values of the corresponding features are extracted and stores in an array as string of integers. For eg, the ruleset for the Load module attack is Protocol=TCP Service=telnet flag=SF dst_host_count=1dst_host_same_port_rate=1is converted into stride length stream as 114111. The pattern for tag 1 is 1211. Now, the dataset is trained with the stride patterns. The 22 different types of attacks are created from the selected dataset and mixed with the original dataset. The tag patterns are fed in to the layered StriFA architecture. The 4 subset of attacks are checked in a layered scheme one layer at a time. Then the mixed dataset is classified as attacks and normal data when the patterns of trained data and test data are matched.

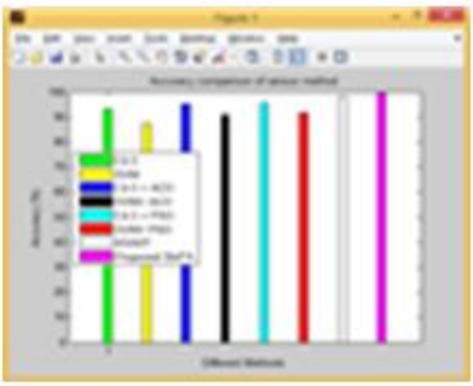


Varying StriFA Architecture

VI. Performance Evaluation

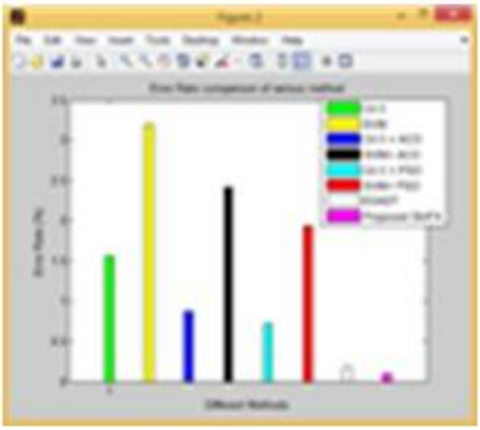
In this section, we compare the performance of our approach with other works in this field. This information is shown in Table.





Method	Accuracy
C4.5	93.23%
SVM	87.18%
C4.5+ACC	95.06%
SVM+ACO	90.82%
C4.5+PSO	91.57%
EDADT	98.12%
Variation of StriFA	99.9071%

Finally, Variation of StriFA took highest accuracy percentage when compared to all six classification based algorithms. Figure 1 specifies the corresponding chart for the result obtained. However, the enhanced StriFA takes less Error rate when compared to other algorithms and provides better accuracy in terms of all other algorithms. Figure 2 specifies the corresponding chart for the result obtained.



VII. Conclusion

In this paper variation of layered StriFA method is carried out on KDD "99 dataset. This is the first step of our research work. As compared to the existing algorithms, the variation of StriFA produced better accuracy with classification of reduced feature set and with less error rate. This proposed method can be improved with the huge volume of dataset and can be analyzed with still more reduced features and new rule sets to detect new attacks.

References

[1]. G.V. Nadiammai, M.Hemalatha "Effective approach toward Intrusion Detection System using data mining techniques", Egyptian Informatics Journal (2014) 15, 37–50

[2]. P Prudhvi, H Venkateswara reddy "Pattern Matching using Layered STRIFA for Intrusion Detection", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014, 6160-6164

[3]. P.Gifty Jeya, M.Ravichandran, C.S. Ravichandran "Efficient Classifier for R2L and U2R Attacks" International Journal of Computer Applications (0975 – 8887) Volume 45– No.21, May 2012

[4]. Safaa O. Al-mamory "Evaluation of Different Data Mining Algorithms with KDD CUP 99 Data Set" Journal of Babylon University/Pure and Applied Sciences/ No.(8)/ Vol.(21): 2013

[5]. Mr. Kamlesh Lahre Mr. Tarun dhar Diwan Suresh Kumar Kashyap Pooja Agrawal "Analyze Different approaches for IDS using KDD 99 Data Set" International Journal on Recent and Innovation Trends in Computing and Communication ISSN 2321 – 8169 Volume: 1 Issue: 8 645 – 651

[6]. Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani "A Detailed Analysis of the KDD CUP 99 Data Set" Proceedings of IEEE Symposium on Computational Intelligence in security and defence applications (CISDA 2009)

[7]. Vivek kshirsagar,Madhuri S.Joshi Rule Based Classifier Models For Intrusion Detection System International journal of Computer science and Information Technology vol(7) 1 2016, 367-370

[8]. Maheshkumar Sabhnani, Gursel Serpen "KDD Feature Set Complaint Heuristic Rules for R2L Attack Detection" EECS Dept, University of Toledo Toledo, Ohio 43606 USA

[9]. Matteo Avalle, Fulvio Risso, "Scalable Algorithms for NFA Multi-striding and NFA-based Deep Packet Inspection on GPUs", IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. X, NO. X, X 2015

[10]. Xiaofei Wang, Yang Xu, Member, "StriFA: Stride Finite Automata for High-Speed Regular Expression Matching in Network Intrusion Detection Systems" IEEE Systems journals Vol.7 No.3 September 2013.

[11]. Giorgio Giacinto,Fabio Roli, and Luca Didaci "Fusion of Multiple Classifiers for Intrusion Detection in Computer Networks"

[12]. Susan M. Bridges, Rayford B. Vaughn "FUZZY DATA MINING AND GENETIC ALGORITHMS APPLIED TO INTRUSION DETECTION" Presented at the National Information Systems Security Conference (NISSC), October 16-19, 2000, Baltimore, MD.

[13]. Xiaofei Wang, Yang Xu, Member, IEEE, Junchen Jiang, Olga Ormond, Member, IEEE, Bin Liu, and Xiaojun Wang