Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 7, Issue, 11, pp.22371-22382 November, 2015

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

ISSN: 0975-833X

RESEARCH ARTICLE

NETWORK INTRUSION DETECTION TECHNIQUE FOR REGULAR EXPRESSION DETECTION USING DPI IN AD-HOC WIRELESS NETWORK

***,[1]Mr. Girish M. Wandhare, [2]Prof. S. N. Gujar and [3]Dr. V. M. Thakare**

[1]Student of Master of Engineering, SKNCOE, Pune, Maharashtra, India
[2]IT Department, SKNCOE, Pune, Maharashtra, India
[3]S.G.B. Amravati University, Amravati, Maharashtra, India

ARTICLE INFO

Article History:
Received 18th August, 2015
Received in revised form
06th September, 2015
Accepted 05th October, 2015
Published online 30th November, 2015

Key words:

Deep Packet Inspection (DPI),
Regular Expression (RegExp),
LaFA, StriFA, CompactDFA,
Tcam, DFA/EC, Deterministic Finite
Automata (DFA), Snort, Bro.

ABSTRACT

Deep Packet Inspection (DPI) is more widely used in all applications or services such as Intrusion Detection System (IDS), which used with or within a network. DPI investigates all data present in the packet as it goes through an inspection to determine the transported application and protocol. Deep packet inspection mostly uses regular expression matching as a main operator. Regular expressions (RegExes) are used to represent complex string patterns flexibly in many applications varying from network intrusion detection and prevention systems (NIDPSs). Regular expressions represent complex string pattern as signatures of attack in DPI. It analyze whether a packet's payload similar to any of a set of predefined regular expressions. There are various techniques implemented in DPI for deep packet inspection for regular expression. We survey on these DPI techniques for further improvement in detection of regular expression in this paper. We implement technique to block regexp packet such as DOS attack and SQL injection attack. In the result we found that it is possible to reduce RegExp transaction memory required to detection in network intrusion detection. We implement this technique with possible use of DPI techniques in the ad-hoc wireless network.

Citation: Girish M. Wandhare, S. N. Gujar and V. M. Thakare, 2015. "Network intrusion detection technique for regular expression detection using dpi in AD-HOC wireless network", *International Journal of Current Research*, 7, (11), 22371-22382.

INTRODUCTION

In most of the applications in real world, Regular expressions (RegExes) are used to flexibly represent complex string patterns in many applications, like network intrusion detection and prevention systems (NIDPSs), DNA multiple sequence alignment and Compilers (Masanori Bando *et al*., 2012). Intrusion detection technique is the process of monitoring the events occurring in a computer system or network and examines them for signs of possible incidents, which are imminent threats of violation of computer security policies and standard of security practices. Intrusion detection and prevention systems (IDPS) are mainly focused on identifying possible type of incidents in packet, logging information about these incidents, try to stop and reporting them to security administrator (Tiwari Nitin *et al*., 2012). Deep Packet Inspection is a technology which enables the network owner to examine internet traffic, throughout the network, in real-time and to separate them according to their payload (Tiwari Nitin *et al*., 2012). Traditional packet investigation algorithms have been limited to match packets to a set of strings.

****Corresponding author: Mr. Girish M. Wandhare,***
Student of Master of Engineering, SKNCOE, Pune, Maharashtra, India.

Newer DPI systems, like Snort (Rafeeq Ur Rehman), and Bro (Bing Chen *et al*., 2006), use rule-sets consisting of regular expressions, these systems are more costly, efficient, and compact in describing attack signatures (Cong Liu *et al*., 2014). In Network intrusion detection system techniques such as Bro (Bing Chen *et al*., 2006), Linux Application Level Packet Classifier (L7 filter) (Xiaofei Wang *et al*., 2013) and Snort (Rafeeq Ur Rehman) use RegExes to represent attack signatures. Regular expressions represent complex string pattern as attack signatures in most of the applications. There is no current regular expression detection system which is capable of supporting large RegEx set; and even larger RegExp sets are expected in future with high speed detection demand (Masanori Bando *et al*., 2012). The LaFA technology implemented on three observations, first RegExes consist of different components like character classes or repetitions, second is the order of components where a RegEx is preserved in the state machine detecting RegEx and third is, many RegExes share similar components (Masanori Bando *et al*., 2012). LaFA requires small amount of memory due to these three contributions: 1) providing specialized and optimized modules detection to increase resource utilization; 2) systematically reordering sequence of the RegEx detection to reduce the number of concurrent operations in detection; 3) sharing states

between automata for different RegExes to reduce requirements of resources.

The TCAM technology is the first hardware dependent RE matching approach that uses ternary content addressable memory (TCAM), It is has been widely deployed in modern networking devices for various tasks such as packet classification. StriFA (Anat Bremler-Barr *et al*., 2014) technology presents the stride finite automata; it's a novel finite automata family class, used to accelerate both string matching and regular expression matching. Compact DFA (Anat Bremler-Barr *et al*., 2014) proposed methodology to compress DFAs by observing that the name used by similar DFA encoding is meaningless. This level of freedom and encode states in such a way that all transitions to a specific state are represented by a single prefix which defines a set of current states. Compact DFA technique applies to a large class of automata that can be categorized by simple properties. With using a TCAM (Chad R. Meiners *et al*., 2014) the throughput of compact DFA reaches to 10 Gb/s with low power consumption in detection. This technique uses Aho–Corasick (AC) algorithm that uses a deterministic finite automaton (DFA) to represent the pattern set (Anat Bremler-Barr *et al*., 2014).

Extended Character set DFA (Tiwari Nitin *et al*., 2012) focused on decreasing the memory storage requirement of DFAs, and it can be partitioned into the following categories: reducing the number of states required, reducing the number of transitions required, reducing the bits encoding the transitions, and reducing the character-set. Unfortunately, these entire approaches compress DFAs at the cost of increased in main memory accesses. This technique propose a novel solution, called deterministic finite automata with extended character-set (DFA/EC) that can significantly reduced the number of states through doubling the size of the character-set. Described solution in this methodology needs only a single main memory access for each byte in the traffic payload (Tiwari Nitin *et al*., 2012).

We implement this project to use deep packet inspection (DPI) techniques for RegExp matching to improve IDS technique in ad-hoc wireless networks. Implementation and comparatively evolution of existing technique with the newly propose technique by considering various parameter such as bandwidth requirement, speed of intrusion detection e.t.c. The implementation details describe in following section.

This project consists of improved RegEx detection technique which will have higher throughput than any other network intrusion detection techniques. We will minimize memory requirements and resource usage by network detection system. The project will be used for regular expression detection of attack signature pattern matching in wireless network. Modules of the projects are survey existing techniques, implement DOS attack filter, built RegEx detection technique which will work on wireless network.

Paper Organization

The remaining content of the paper organized as follows. An overview of deep packet inspection is described in Section 3. Here we describe Detail working of DPI and levels of DPI. Section 4 includes the use of regular expression in DPI. Section 5 gives limitations related to DPI techniques for RegExp detection. Ad-hoc network describes in section 6. Dos attacks and their types describes in section 7. Our proposed system details and implementation describe in section 8. Section 9 describes advantages of our proposed system. Result and comparison of our technique with existing system include in section 10.In section 11 we conclude our work and contains our future work.

Literature survey and elaboration & synthesis

3.1 Deep Packet Inspection

Deep Packet Inspection (DPI) is a technology that allows the network owner to analyses internet traffic, transmitted through the network, in real-time and to differentiate traffic on the basis of their payload. Originally the Internet protocols need the network routers to scan only the header of an Internet Protocol (IP) packet. The packet header includes the origin and destination address and other information about to moving the packet across the network. The "payload" or content of the packet that contains the text, images, files or applications send by the user, was not considered to be a concern of the network operator. DPI enables network operators to scan the payload of IP packets as well as the header. Figure 3.1 (Klaus Mochalski and Hendrik Schulze, 2008) shows the domain of packet inspection required in internet protocols and in DPI (Klaus Mochalski and Hendrik Schulze, 2008).

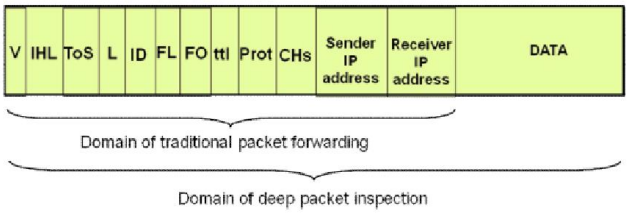


Figure 3.1 Domain of Deep Packet Inspection (Klaus Mochalski and Hendrik Schulze, 2008)

DPI system uses regular expressions to define patterns of interest in network data flow streams. The equipment is programmed to take decisions about how to handle the packet or a stream of packets according to the recognition of a regular expression or pattern in the payload of network.

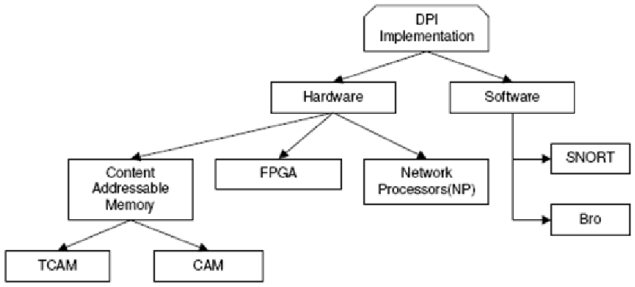


Figure 3.2 DPI Implementation (Tamer AbuHmed *et al*., 2008)

This enables networks to differentiate and control traffic on the basis of the applications, content, and subscribers (Klaus Mochalski and Hendrik Schulze). Figure 3.2 shows the DPI implementation in software and hardware modules.

3.1.1 Regular Expression

Deep packet inspection mostly uses regular expression (RE) matching as a main operator. It analyze whether a packet's payload matches any of the combination of predefined regular expressions. REs are fundamentally more costly, efficient, and flexible in specifying attack signatures. Earlier RE matching algorithms are either software base or field-programmable gate array (FPGA) based (Masanori Bando *et al*., 2012).

RegExes includes a variety of different components like character classes or repetitions (Masanori Bando *et al*., 2012). Because of this variety, it is hard to a method identification that is efficient for concurrently detection of all these different components of a RegExp. Most of the RegExes share similar components. In the traditional Finite Automata, a small state machine is used to identify a component in a RegExp. This state machine is duplicated since the similar component may appear multiple times in different RegExes. Furthermore, most of the time, RegExes share these components that cannot appear at the same time the input. As a result, the repetition of the same state machine is for different RegExes result in redundancy and limits the scalability of the RegEx detection system. Figure 3.3 shows example illustrating the transformation from a RegEx set R into the corresponding LaFA technique (Masanori Bando *et al*., 2012).

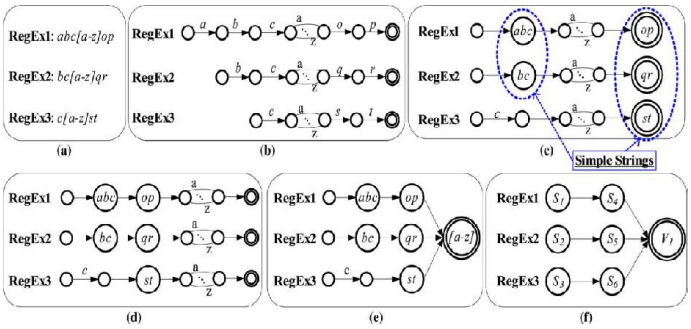


Figure 3.3 Example illustrating the transformation from a RegEx set R into the corresponding LaFA. (a) RegEx set. (b) NFA corresponding to. (c) Separation of simple strings. (d) Reordering of the detection sequence. (e) Sharing of complex detection modules. (f) LaFA representation of the RegExes (Masanori Bando *et al*., 2012)

At present, regular expressions are replacing explicit string patterns by the pattern matching language of choice in packet scanning applications. Their worldwide use is due to their expressive power and flexibility for describing useful patterns. For example, in the Linux Application Protocol Classifier (L7-filter), all protocol detectors are expressed as regular expressions. Likely, the Snort intrusion detection system has evolved from no regular expressions in its rule set in April 2003 to 1131 out of 4867 rules uses regular expressions as of February 2006. Another intrusion detection system, Bro (Bing Chen *et al*., 2006), also use regular expressions as its pattern language (Fang Yu *et al*., 2006).

A regular expression expresses a set of strings without enumerating them explicitly. Table 3.1 show lists of the similar features of regular expression used in packet payload scanning. For example, consider a regular expression string from the Linux L7-filter for detection of Yahoo traffic: "^(*ymsg*|*ypns*|*yhoo*).?.?.?.?.?.?.?(*lwt*).*\x*c0*\x*80*". This pattern matches any packet payload that starts with *ymsg, ypns,* or *yhoo*, followed by seven or fewer arbitrary characters, and then a letter *l, w* or *t*, and some of arbitrary characters, and at last the ASCII letters *c0* and *80* in the hexadecimal form (Fang Yu *et al*., 2006).

Table 3.1 Features of Regular Expressions (Fang Yu *et al*., 2006)

Syntax	Meaning	Example
^	Pattern to be matched at the start of the input	^AB means the input starts with AB. A pattern without '^', e.g., AB, can be matched anywhere in the input.
I	OR relationship	*A*|*B* denotes *A* or *B*
.	A single character wildcard	
?	A quantifier denoting one or less	*A*? denotes *A*, or an empty sting.
*	A quantifier denoting zero or more	*A** means an arbitrary number of *A*s.
{}	Repeat	*A*{100} denotes 100 *A*s.
()	A class of characters	(*lwt*) denotes a letter *l*, *w*, or *t*.
(^)	Anything but not n	(^\n) denotes any character except \n.

3.1.2 Deterministic Finite Automata (DFA)

The Deterministic Finite Automata (DFA) contains a finite set of input symbols (which are denoted as P), a finite set of states, and a transition function to move from one state to the other state denoted as @. Compared to NFA, DFA has only one active state at a time (Cong Liu *et al*., 2014; Fang Yu *et al*., 2006).

The regular expression is required as a need for packet payload analysis to different protocols packets. It introduces limited DPI techniques to deal with all network packets structures. Because of this limitation, state-of-art systems have been introduced to substitute the string sets of intrusion signature with expressiveness regular expression (regexp) systems. Hence, there are several content inspection engines that have partially or completely migrated to regexps including those in Snort (Rafeeq Ur Rehman), Bro (Bing Chen *et al*., 2006), and Cisco systems'.

Experimentally, DFA for regexp which contains hundreds of pattern yields to tens of thousands of states that mean memory consumptions in large number of megabytes. As a solution of one of the common problems of hardware based DPI solution that is the memory access because the memory accesses for the contents of the off chip memory are proportional with the number of bytes in the packet (Fang Yu *et al*., 2006).

3.2 Related Work

3.2.1 LaFA (Masanori Bando *et al*., 2012)

LaFA is a novel detection technique that resolves scalability issues of the present RegEx detection paradigm. LaFA is finite

automata improved for scalable RegEx detection. LaFA is used for representing a set of RegExes (R={r1;r2;r3;…}) can be also called a *RegEx database*. An associated LaFA RegEx detection system can be queried with an input stream, such as a network packet. The system will result *a match* along with a list of matching RegExes if input includes one or many of the RegExes in. Else, a *no match* is returned. LaFA simplify the reduction of memory requirements and detection complexity.
LaFA architecture shown in following Figure 3.4 contains two blocks, the detection block and the correlation block Detection Block is responsible for detection of the component in the input packet string. The simple string optimized for exact string matching module for detection of simple string.

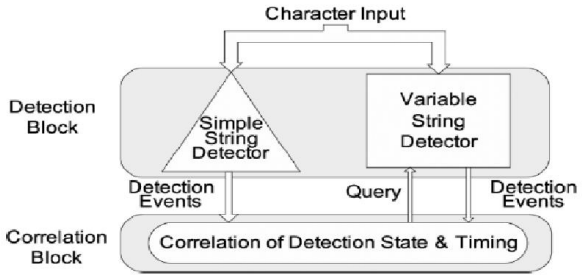


Figure 3.4 LaFA Architecture (Masanori Bando *et al*., 2012)

The variable string detector contains highly optimized modules for variable string detection. The correlation block used to track RegEx status. The correlation block examines the sequence of components in the input string (i.e., order and timing). The detectors in the detection block communicate their findings to the correlation block by exchanging detection events. Each detection event consists of a unique detected component ID and the input packet location.

3.2.2 Small TCAM (Chad R. Meiners *et al*., 2014)

Small TCAM is the first RE matching solution based on ternary content addressable memory (TCAM). This methodology uses a TCAM and its related SRAM to encode the transitions of the DFA develop from an RE set where one TCAM entry might be encode multiple DFA transitions. There are three key reasons why TCAM-based RE matching does well give in this research: a small TCAM is adequate of encoding a large DFA with carefully designed algorithms. TCAMs facilitate less time RE matching because TCAMs are essentially high-performance parallel lookup systems. TCAMs are off-the-shelf chips which are universally deployed in advance networking devices; it should be easy to design networking devices that includes TCAM-based RE matching solution (Chad R. Meiners *et al*., 2014). In this project, TCAM capable to store a DFA with 25 K states in a 0.5-Mb TCAM chip; most DFAs need at most one TCAM entry per one DFA state. With variable striding it shows a result of up to 18.6 Gb/s is possible.

3.2.3. StriFA (Xiaofei Wang *et al*., 2013)

StriFA methodology has been implemented in software and compared based on different traces. It accepts the problem of developing a variable-stride pattern matching engine that can achieve high matching speed with a relatively less memory usage. The technology comes up with stride finite automata (StriFA) that can process a number of characters at a time. StriFA is implemented to be immune to the memory blow-up and byte alignment problems, and therefore, it requires much less memory compare to the previous schemes. Stride deterministic finite automaton (StriDFA) and stride nondeterministic finite automaton (StriNFA) are two basic types of development of StriFA.

The technology results showed that this methodology can achieve about 10 times increase in speed, with a lower memory requirement compared to traditional NFA/DFA, while maintaining the same detection capabilities.

3.2.4 CompactDFA (Anat Bremler-Barr *et al*., 2014)

The main advantage of CompactDFA is that it capable to fits into commercially available IP-lookup solutions. Also they may be used for performing fast pattern matching. The output of the CompactDFA technique is a set of compressed rules, such that there is only a rule per state. In the compressed set of rules, a code of a state may match multiple number of rules. CompactDFA gives the intuition behind each of its three stages: State Grouping, Common Suffix Tree Construction, and State and Rule Encoding.

CompactDFA used only for the pattern matching process. It uses two similar pattern sets: Snort and ClamAV. This methodology can achieve high speed pattern matching of 2 Gb/s with small amount of power consumption. This technique describes a reduction of the pattern matching problem to the IP-lookup problem. Because of its small memory and low power requirements, this architecture can be implementing with many TCAM working in parallel. Each TCAM match pattern on a different session, achieving a total throughput of 10 Gb/s and more.

3.2.5 A DFA with Extended Character-Set (Cong Liu *et al*., 2014)

This techniques implementation based on general-purpose processors that are cost-effective and flexible to update. It proposes a novel solution, known as deterministic finite automata with extended character-set (DFA/EC), which can significantly reduce the number of states by doubling the size of the character-set. Solution describes in this methodology need only a single main memory access for each byte in the traffic payload. It performs experiments with number of Snort rule-sets. Results show that DFA/ECs are very compact and are over four orders of magnitude smaller in the best cases compared to DFA.

The advantages of a DFA/EC are described in the following: A DFA/EC need only one main memory access for each byte in the packet payload, while significantly decrease storage in terms of table size. A DFA/EC is conceptually easy to implement, and simple to update due to fast speed construction. It maintains two states in runtime: one state represent the DFA, and an additional state for the supporting program. The running time state of the main DFA is denoted by a DFA state label,

and state of the supporting program is denoted by a set that includes currently active complementary states. For every byte in the payload, the DFA/EC functions given as follows. Firstly, the complementary program evaluate the extra bit for the extended character with the help of the next byte and the current state of the complementary program and the next state of the DFA and a label is looked-up by using the current state of the DFA and the extended character that is composed of the next byte and the extra bit. Then the complementary program calculates its next state by using its current state, the next byte, and the label on the main DFA transition in this implementation, the transition functions of the main DFA, is built by a transition table; and is implemented by the complementary program that only contains several efficient instructions.

The requirement of total minimum memory (storage) for the transition tables in terms of bits and the number of bits is the multiple of the number of transitions and the number of bits needed to encode each transition evaluated. It describes result as; the memory bandwidth of DFA/EC can even be smaller than DFA in rule-sets exploit-19 and web-misc-28

3.2.6 Dos Attack

A Distributed Denial of Service (DDoS) attack is a tray to make an online service unavailable by overloading it with traffic from multiple sources. They target a wide class of important resources, from banks to news websites, and produce a major challenge to making sure people can publish and access important information.

Attackers develop networks of infected computers; known as 'botnets', by distribute malicious software via emails, websites and social media. Once get infected, these machines can be controlled remotely, without their owner's knowledge and permission, and used like an army to execute an attack against any target. Botnets can create huge floods of traffic to overwhelm a target. These floods can be develop in multiple ways, like sending more connection requests than a server can handle, or having computers send the client huge amounts of random data to use up the target's bandwidth. Some attacks are so huge they can max out a country's international cable capacity.

Difference between DOS and DDOS Attack

It is important to differentiate among Denial of Service (DoS) attacks and Distributed Denial of Service (DDoS) attacks. In a DoS attack, a single computer, a single internet connection is used to flood a server with requests packets, with the goal of overload the targeted server's bandwidth and resources. In DDoS attack many devices and multiple Internet connections distributed globally into what is known as a botnet. Hence a DDoS attack is much harder to deflect, simply because there is no single attacker to secure from, as the targeted resource will be overloaded with requests from many hundreds and thousands of multiple sources.

3.2.7 XSS and SOL Injection Attack

XSS Attack: XSS attacks are Cross-Site Scripting attacks. These attacks are a type of injection attack, in which contaminated scripts are injected into benign and trusted web sites. XSS attacks attack when an attacker uses a web application to send malicious scripting code to a different end user. Flaws which allow these attacks to succeed are quite widespread and start anywhere a web application get input from a user within the output it create without validating or encoding it.

An attacker may use XSS to send a malicious script code to an unsuspecting user. The end user's browser does not way to know that the script could not be trusted, and will execute the script. Because of it thinks the script came from a trusted source, the malicious script code can access any cookies, session tokens, or other sensitive information retained by the browser from victim's machine. These scripts can even rewrite the content of the HTML page.

XSS attacks may be conducted without using <script></script> tags. Other tags will do exactly the same thing, for example:

```
<body onload=alert('test1')>
```

or other attributes like: onmouseover, onerror.

Onmouseove

```
<b onmouseover=alert('Wufff!')>click me!</b>
```

Onerror

```
<img src="http://url.to.file.which/not.exist"
onerror=alert(document.cookie);>
```

SQL Injection Attack

SQL injection attack is a technique by which malicious users can inject SQL commands into an SQL statement through web page input. Injected SQL commands can modify SQL statement and trespass the security of a web application. Assume that the main purpose of the code was to create an SQL statement to select a user by a given user id. If there is nothing to restrict a user from entering "wrong" input, the user can enter some "smart" input like this:

UserId:

105 or 1=1

Server Result

SELECT * FROM Users WHERE UserId = 105 or 1=1

The SQL query above is valid. It will return all rows from the table Users, since **WHERE 1=1** is always true. The example above looks dangerous, What if the Users table contains usernames and passwords?

The SQL statement above is much the same as this:

SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1

A smart hacker might be get access to all the user names and passwords in table of a database by simply inserting 105 or 1=1 into the input textbox.

Problem with Existing Systems

From the survey of above techniques we found that, the approach describe in these techniques may require a large

number of transitions for some cases, leading to an increase in the number of memory accesses per input byte. In addition, DFA construction is complicated and requires significant resources (Masanori Bando *et al*., 2012). There is very few network intrusion detection techniques discover in wireless ad-hocnetworks.

CompactDFA methodology used in architecture requires several TCAM working in parallel because of its small memory and lower power requirements. NBA technologies have some significant restrictions. They are delayed in removing attacks due to their data sources, especially when they depend on flow data from routers and other network devices (Tiwari Nitin *et al*., 2012). DFA/EC does not merge with the existing transition compression techniques and character-set compression techniques, and perform experiments with many rule-sets (Cong Liu *et al*., 2014). One of the problems for StriFA is choosing an appropriate tag. Since in both the rules and the incoming traffic, the probabilities of different characters occurrence vary from each character to other, it is a problem to select an appropriate tag from the rule set (Xiaofei Wang *et al*., 2013).

Following table shows comparison of existing network intrusion detection techniques.

Table 4.1 Comparison of deep packet intrusion techniques

Intrusion Detection Techniques	Throughput
LaFA	34 Gb/s
CompactDFA	10 Gb/s
Small TCAM	18.6 Gb/s
StriDFA	26.5 Gb/s

4.1 Objectives of Problem

Following are objectives of problem we define on the basis of above survey:

1. We can improve network intrusion detection throughput with use of DPI techniques.
2. LaFA technique can be modifying for effective detection of evaluating RegExp on the network.
3. Higher throughput in network intrusion detection can be possible.
4. Above discuss techniques could have better performance in memory requirements, speed of detection, detection of evaluating RegEx detection.
5. There is very few intrusion detection techniques work on wireless network.

System Implementation

5.1 Module Partitioning

This section we include the partitioning of project into different modules. All these modules are explained as follows:

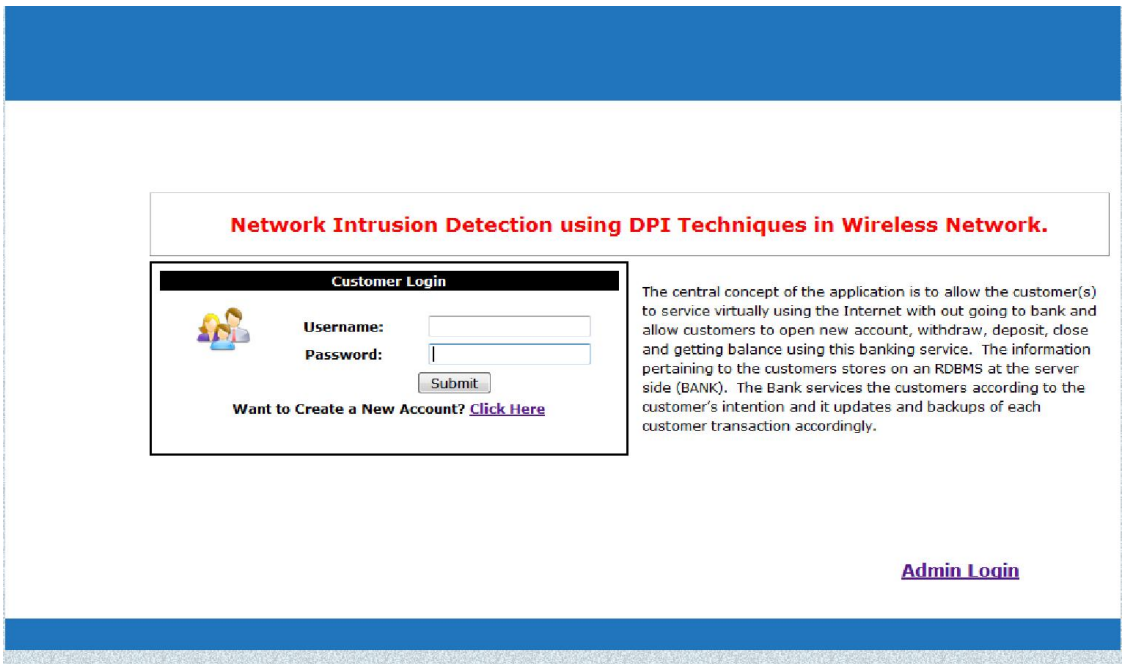
5.1.1 Client and Server

This module includes the registration and log in of client and server. It includes following classes.

- Log In : With this class client can login to the server web site i.e. Banking website
- Register: this class used to register new user for server site.
- Admin Login: Admin can log in and manage the data of website.
- Admin View Log: With this class admin can view all the block IP and type of attacks done and its date and time.
- Unblock: With this class admin can unblock the block IP.

5.1.2 Client side snapshot

Log in Page



User Account Page Snapshot

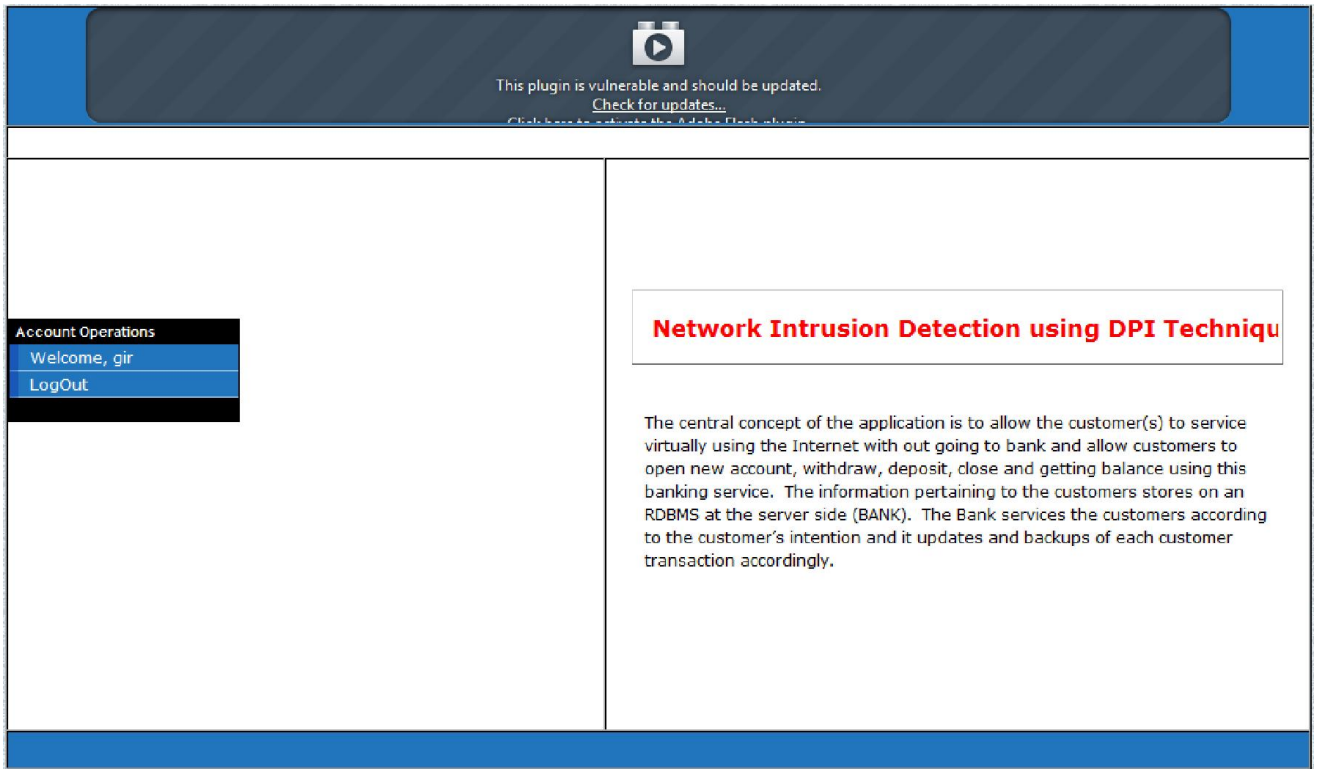


Figure 5.2 User Account Snapshot

Admin Login Page

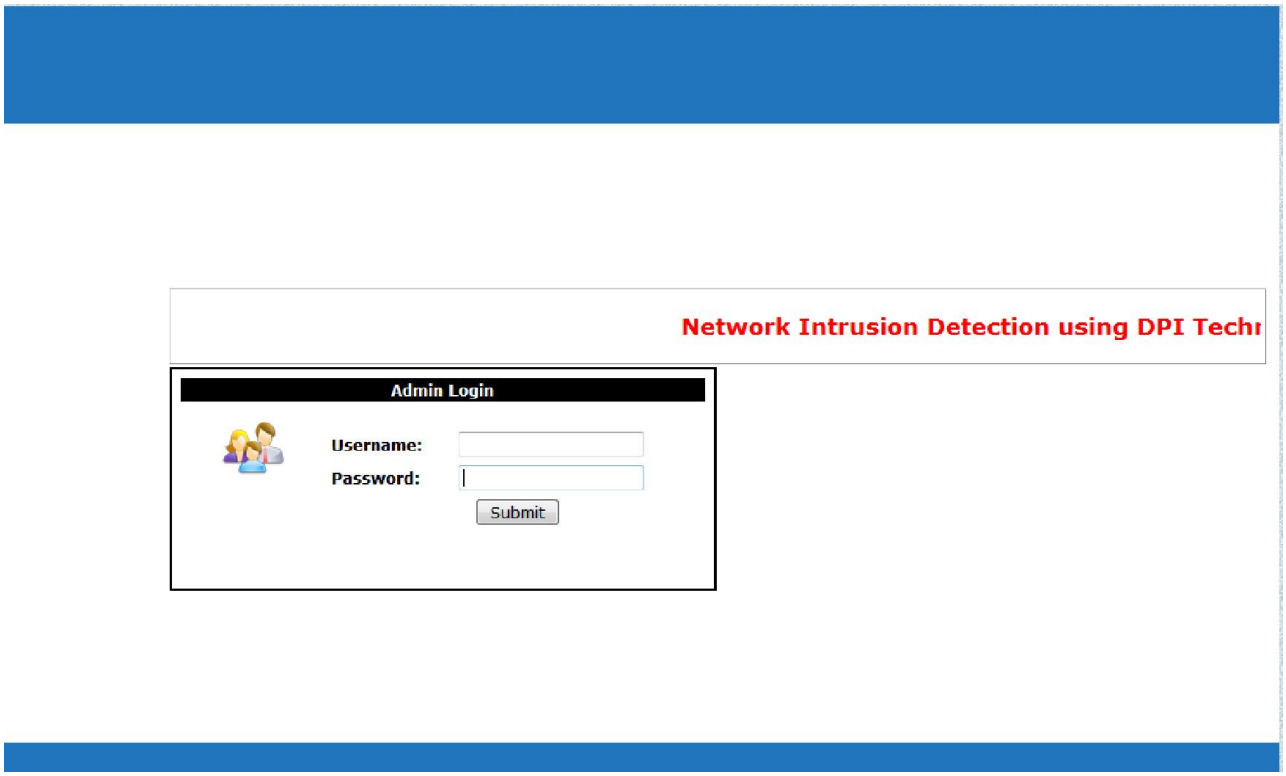


Figure 5.3 Snapshot for admin log in page

Admin Account Page

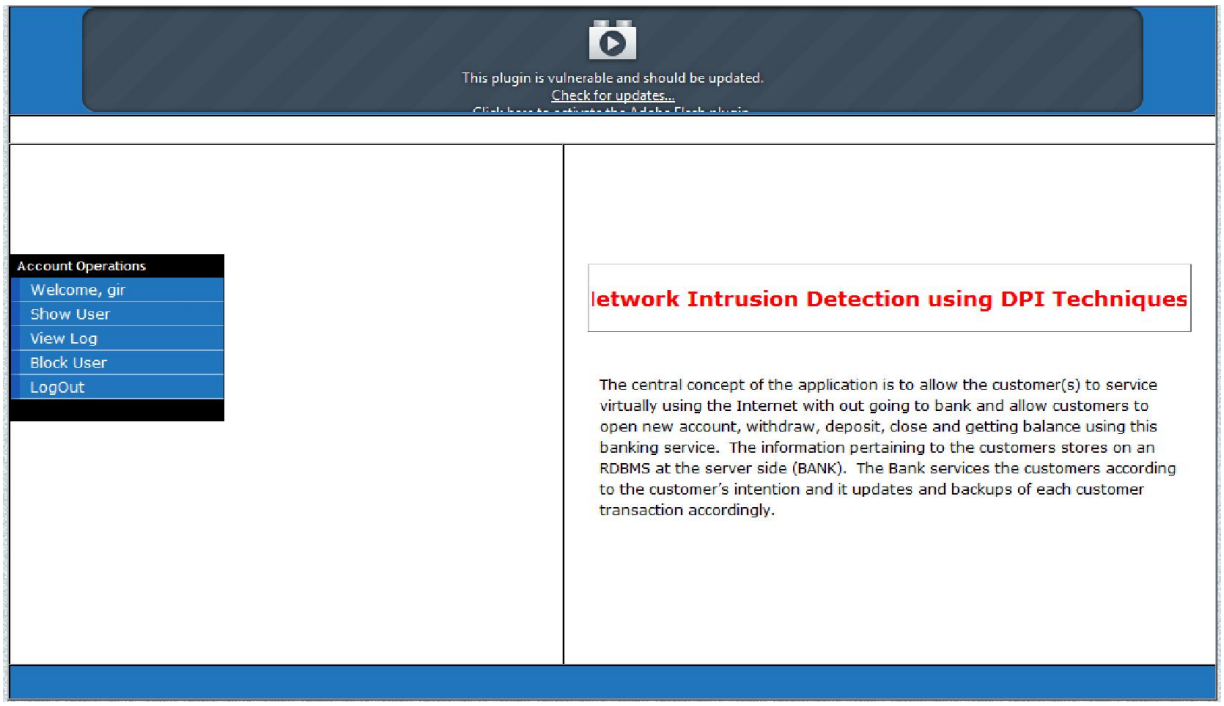


Figure 5.4. Snapshot for admin log in page

5.1.3 DDOS Attack Filter

- Request Count: This class count no. of requests per second (RPS) coming from client.
- RPS Limit: This class used to save the limit of the request per second any human user can make.
- Block IP: DDOS filter block the user IP if RPS exceeds the limit.

5.1.4 DPI module

- Pattern Matching: This class used deep packet inspection pattern matching algorithm to detect the attack pattern.
- Block IP: If request packet pattern match to attack pattern saved in database then it block the client IP.

View Log Page

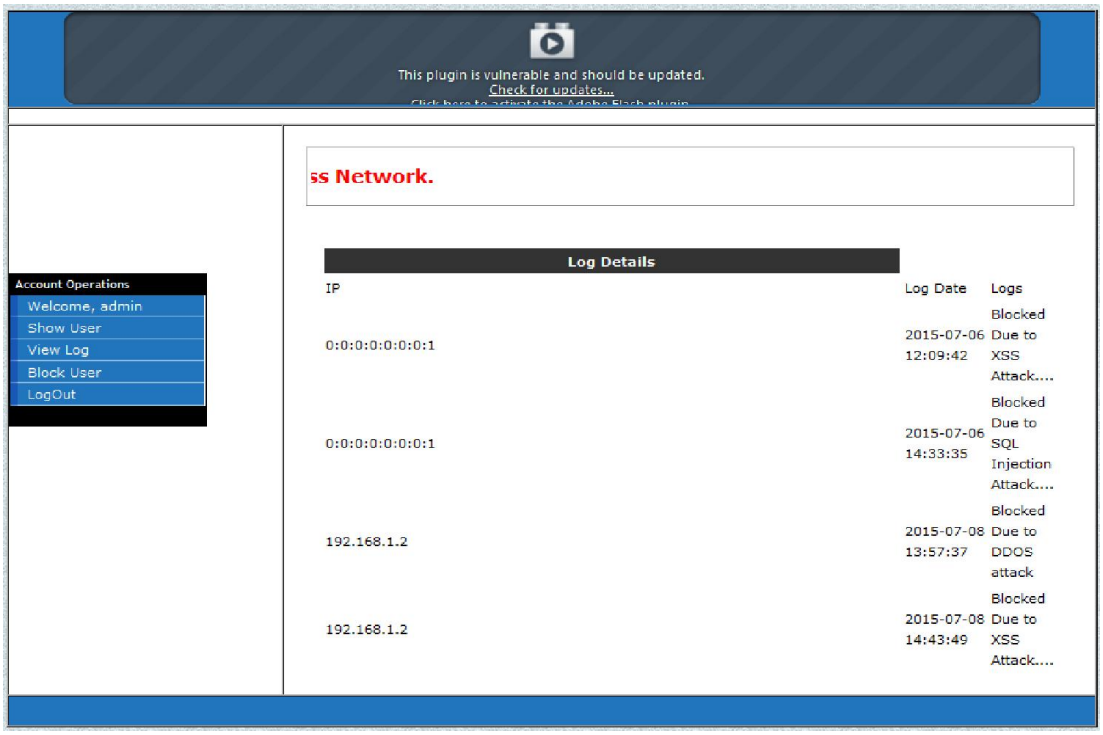


Figure 5.5 Snapshot of attack log detail page

Block IP Page Snapshot

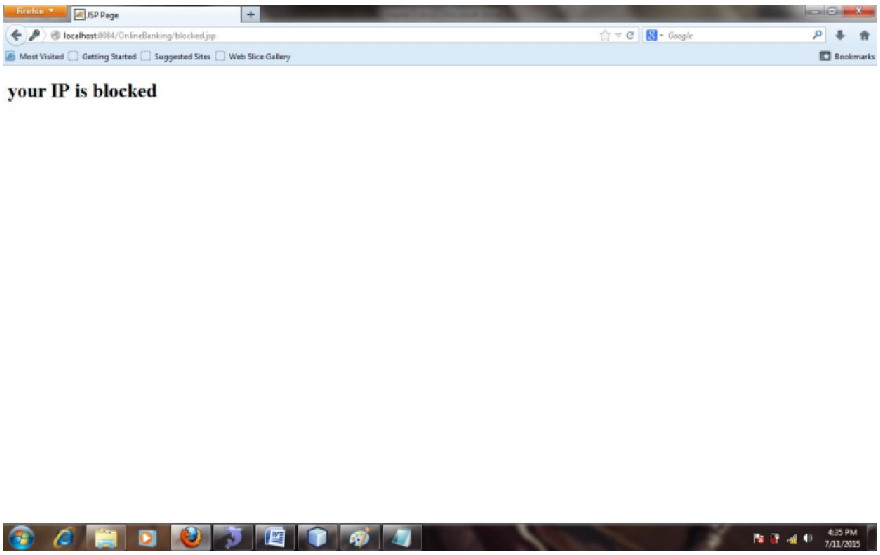


Figure 5.6 Snapshot of Block IP Page

SQL Injection Attack Snapshot

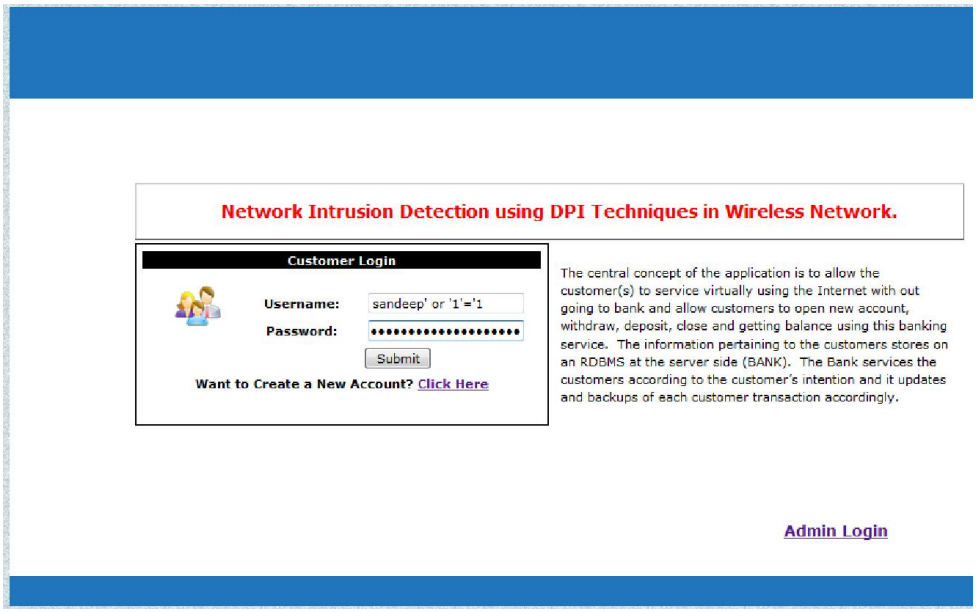
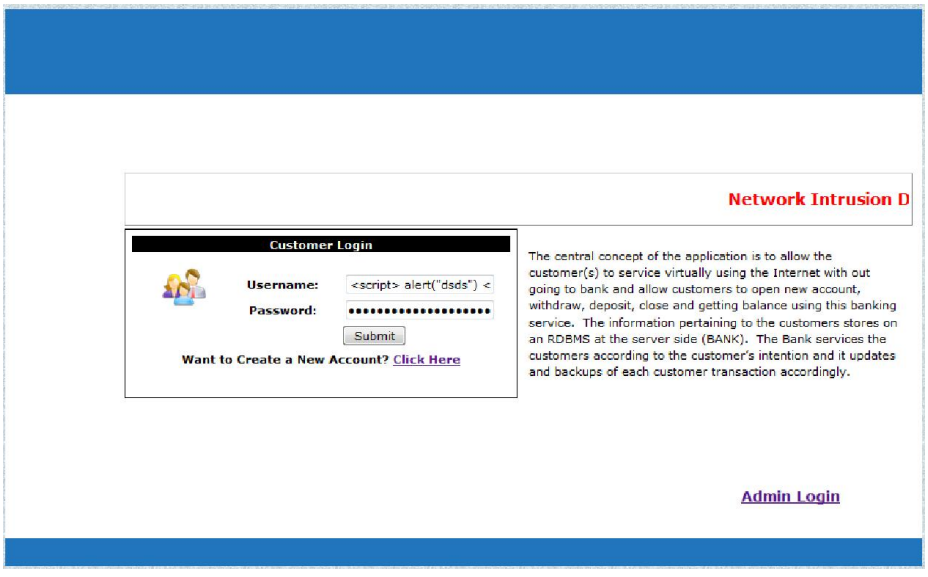


Figure 5.7 Snapshot of SQL Injection Attack

XSS Attack Snapshot



5.1.5 Database

- In this project we use Apache Tomcat sever to access database named IDS.
- This database used to save attack pattern and request count per second and access by DDOS filter and DPI module to search for the attack pattern.
- Database used to store account data, user account id and password, list of blocked ip, type and date and time of attack when it occurs.

5.2 Low-level design

Low-level designs of software system include:

- Private classes, private methods, private attributes
- Algorithms.

Low-level design also provides an interface for all classes, public and private methods, including parameters, return values, exceptions thrown and types defined. It describes and justifies the choice of data structures, describes the major alternatives that are considered and why the choice is preferred that is opted.

In our project private classes such as block ip class, account information class only can be access by owner. Owner can login to the server using admin login and see the all information from the site. Admin also have privileges to alter the data saved in database such as unblock the user ip. Client can only access the information about his account, hr does not see any attack information or other user account data. If client try to make attack on the server it can block by dos filter and DPI module.

DOS filter compare the request count per second with the possible no. of the request any human user can make which is saved in database and make the decision to block the ip or not. DPI used to compare packet pattern with the attack pattern saved in database if attack pattern match then it blocks the user op who try to attack on server.

5.3 Implementation Setting

We implement our project on Net beans IDE 7.3. Project is implemented in java language. We used SQL yog enterprises to use Apache Tomcat server. Client server model is used to run this project. To connect server and client we implement Ah-hoc wireless connection between them. We also test our project on internet by connecting internet connection. Data owner & authorized user can login through any machine.

Mathematical Model of Problem Statement

Mathematical Model for simple regular expression matching
Consider
R=regular expression
R= abc(a-z)op
P=pattern with which we match the input packet regular expression.
If
R=P (Regular expression pattern match)
i.eR(i)=P(i)
R(ii)=P(ii)
R(iii)=P(iii) . . . . .
Then the packet blocked
Else the packet forward to the server.
In our approach we divide regular expression in two division
S= set if simple variables in regular expression.
C= set of complex variables in regular expression.
Consider
R= adc(a-z)op
We divide this regular expression
S= abcop
C= (a-z)
If
S=P (Pattern Matching)
C=P (Pattern Matching)
Then the packet is block
Otherwise packet forwarded to the server.

Figure 6.1 shows RegEx components based on their detection complexity and depth.

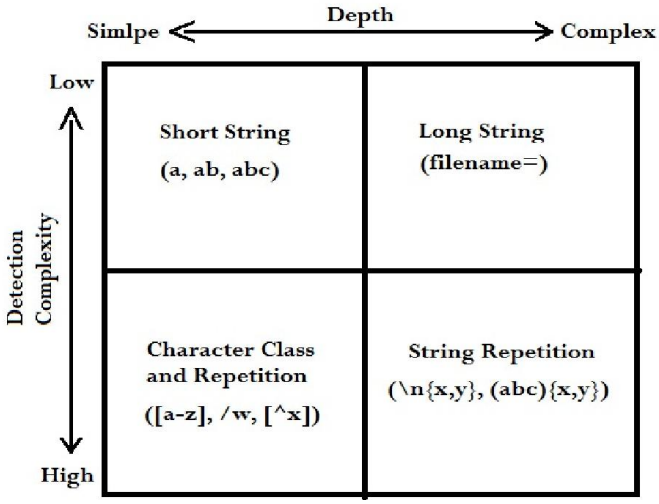


Figure 6.1 RegEx components based on their detection complexity and depth

6.1 Mathematical Model For dos filter

In DDos filter we count number of request per 40 seconds from one IP address, if the no requests are greater than 100 then we block the IP address. Thus any human cannot send hundred requests per 40 second.

Suppose

r = request count per 40 seconds (time stamp) from particular IP.
t= time stamp which we set 40 seconds.
c = no of request any human can possibly send to server here we set (100).
If
r > c in time t
Ip address blok, (No further request from this IP processed)
If not

Then allow IP to communicate with server.

6.2 Architecture Diagram

Figure 6.2 shows architecture diagram for our project. It shows client and server transaction which done through DOS filter and DPI module to avoid attack on server such as DOS attach and Sql Injection Attack.

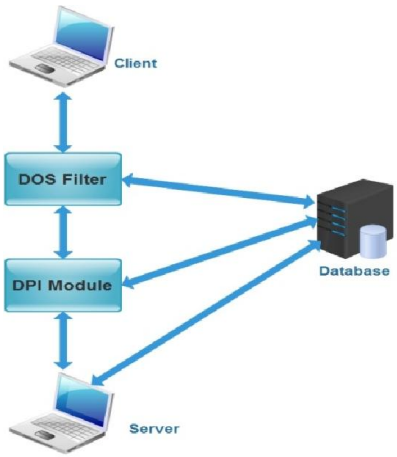


Figure 6.2. Architecture Diagram

6.3 ER Diagram

ER Diagram means Entity Relationship Diagram. The Entities are mapped to the tables in the application. An entity-relationship (ER) diagram is a UML diagram that shows relationships between entities in a database. In figure 6.3 client, DOS filter, DPI module, server and data base are entities. Connection line shows the relationship between them, and ovals represent attributes of entities.

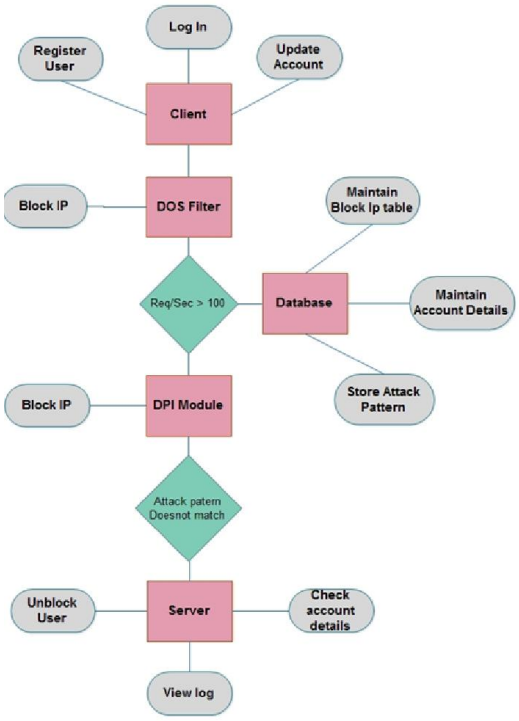


Figure 6.3 E-R diagram for our project

6.4 DFD Diagram

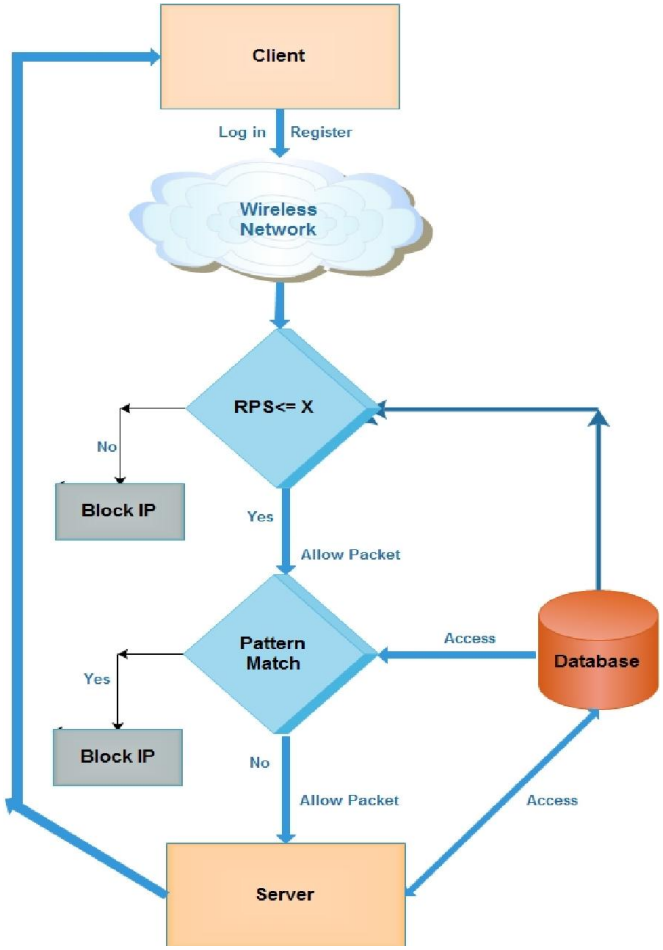


Figure 6.4 DFD level 2 Diagram (Dpi Module)

7. Test Procedures Results and Discussion

7.1 Test Plan

Table 7.1. Test Plan for Project

S. No	Content	Explanation
1.	Name of the product	Network Intrusion Detection Using Dpi Techniques In Wireless Network.
2.	Prepared By	Girish M. Wandhare
3.	Introduction	Testing is carried out mainly for finding all defects present in the system and to prevent a defective product reaching the customers. Testing is also carried out to convince the customer that product is fulfilling the specifications and functional requirement of customer.
4.	Objectives	To test all the functionality of a module. Test correctness of input and output. Test the component point of view estimation accuracy.
5.	Scope	Test the functionality for traditional development and Component based software development.
6.	Testing Strategy	Unit Testing Module Testing Performance Testing User acceptance Testing Beta Testing.
7.	Approval	Prof. S. N. Gujar (Assistant Professor), SKNCOE

7.2 Advantages of Proposed System

With this proposed methodology improve network intrusion detection throughput with use of DPI techniques. The Dos filter remove or blocked all dos attack malicious packet it filters the dos attack packets due to which it improve malicious packet detection.

- Speed of intrusion detection and prevention is increased.
- Authorized user can access data without any security risk.
- Memory requirement for server security is reduced.

7.3 Result and analysis

In this section we discuss on the result of our project. After all the test cases we test our program for various DDOS attack on our server site web page. It results in machine IP block as we expected. We test our project for Sql injection and XSS attack for testing DPI technique pattern matching by inserting various Sql injection and XSS attack, our project blocked every attack. We also tested admin privileges to unlock any client which would be block earlier. Admin can unblock any client from the database by accessing admin login.

Admin can also see the information about attack that what kind of attack occurs at which date and time, Server can store the list of attack according to its time and date.

Conclusion and Future Enhancement

In existing systems, there are some limitations Intrusion detection and prevention system. In our project DDOS filter blocks the DDOS attacks so that the intrusion detection module only face packet without DDOS attack pattern, hence it automatically increase intrusion detection and prevention speed.

In DDOS filter we implement technique to count request per second and set the value which cannot be access by any human user, If the request per second count is greater than predefined value system block clients ip, so that the attacker cannot try to attack on server again. Intrusion detection module use Deep Packet Inspection pattern matching technique to detection of intrusion containing packets and prevent it.

We implement our project with the use of wireless ad-hoc network; we tested it with this network and internet as client server module.

8.1 Future Enhancement

In future we can improve intrusion detection by using various emerging attack patterns in intrusion detection module.

****

REFERENCES

Anat Bremler-Barr, DavidHay, and Yaron Koral, "CompactDFA: Scalable Pattern Matching Using Longest Prefix Match Solutions", *IEEE/Acm Transactions On Networking*, Vol. 22, No. 2, April 2014.

Bing Chen, Lee, J., and Wu, A.S., "*Active event correlation in Bro IDS to detect multi-stage attacks*", Fourth IEEE International Workshop on Information Assurance, 13-14 April 2006.

Chad R. Meiners, Jignesh Patel, Eric Norige, Alex X. Liu, and Eric Torng., "*Fast Regular Expression Matching Using Small TCAM*", IEEE/Acm Transactions On Networking, Vol. 22, No. 1, February 2014.

Cong Liu, Yan Pan, Ai Chen, and Jie Wu., "*A DFA with Extended Character-Set for Fast Deep Packet Inspection*", IEEE Transactions on Computers, Vol. 63, No. 8, August 2014.

Fang Yu, Zhifeng Chen, Yanlei Diao, T. V. Lakshman, and Randy H. Katz, "*Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection*", ACM 580-0/06/0012, December 3–5, 2006.

Klaus Mochalski, and Hendrik Schulze, "*White paper on Deep Packet Inspection*", ITU-T study groups com13.

Masanori Bando, N. Sertac Artan, and H. Jonathan Chao., "*Scalable Lookahead Regular Expression Detection System for Deep Packet Inspection*", IEEE Transactions on Networking, Vol. 20, No. 3, June 2012.

Rafeeq Ur Rehman, "*Intrusion Detection Systems with Snor*t", ISBN 0-13-140733-3, Library of Congress Cataloging-in-Publication Data, Prentice Hall PTR Upper Saddle River, New Jersey 07458.

Tamer AbuHmed, Abedelaziz Mohaisen, and DaeHun Nyang., "*A Survey on Deep Packet Inspection for Intrusion Detection Systems*", Information Security Research Laboratory, Inha University, Incheon 402-751, Korea, March 2008.

Tiwari Nitin, Solanki Rajdeep Singh and Pandya Gajaraj Singh, "*Intrusion Detection and Prevention System (IDPS) Technology- Network Behavior Analysis System (NBAS)*", *ISCA Journal of Engineering Sciences*, Vol. 1(1), 51-56, July 2012.

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*", *IEEE Systems Journal*, Vol. 7, No. 3, September 2013.