Internet Traffic Classification by Aggregating Correlated Naive Bayes Predictions

[1]Sandhya R. M., [2]Pramod M. Murari, [3]Shanoor. M. Mayannavar, [4]S. G. Gollagi, [5]Y. M. Naik

[1,3,5]Assistant Professor, [2]Asst. Prof.(E&E), [4]Associate Professor
Department of Computer Science and Engineering
Hirasugar Institute of Technology
Belagavi, India

***Abstract*—with the current advancement in the usage of online applications and services, the concept of network security has fascinated many researchers particularly in the field of intrusion detection. The private data being sensitive if manipulated or intruded leads to severe issues resulting in disruption of the services. This demands to take required preventive measures to safeguard the sensitive information that flows through the internet traffic. Thus traffic that regularly flows through to traffic needs to be analyzed and classified to identify the patterns that might lead to denial of service or attacks or worm propagations etc. Here we use naive Bayes (NB) classification methods applied in Internet traffic classification, which is a simple and effective probabilistic classifier employing the Bayes' theorem with naive feature independence assumptions This paper proposes a mechanism for false positive/negative assessment with multiple IDSs/IPSs to collect FP and FN cases from real-world traffic and statistically analyze these cases.**

***Keywords*— Naive Bayes (NB) classification,quality of service (QoS) control, traffic classification, feature discretization, intrusion detection and prevetion system, probabilistic classifier**

1. INTRODUCTION

Application oriented traffic classification is a fundamental technology for modern network security. It is useful to tackle a number of network security problems including lawful interception and intrusion detection. For example, traffic classification can be used to detect patterns indicative of denial of service attacks, worm propagation, intrusions, and spam spread. In addition, traffic classification also plays an important role in modern network management, such as quality of service (QoS) control. While traditional traffic classification techniques may rely on the port numbers specified by different applications or the signature strings in the payload of IP packets, modern techniques normally utilize host/network behavior analysis or flow level statistical features by taking emerging and encrypted applications into account.

Recent research shows that flow statistical feature based traffic classification can be enhanced by feature discretization. Particularly, feature discretization is able to dramatically affect the performance of naive Bayes (NB).
An intrusion detection system (IDS) is a device or software application that monitors network or system activities for malicious activities or policy violations and produces reports to a Management Station. Some systems may attempt to stop an intrusion attempt but this is neither required nor expected of a monitoring system. Intrusion detection and prevention systems (IDPS) are primarily focused on identifying possible incidents, logging information about them, and reporting attempts. In addition, organizations use IDPS for other purposes, such as identifying problems with security policies, documenting existing threats, and deterring individuals from violating security policies. IDPS have become a necessary addition to the security infrastructure of nearly every organization. False positives and false negatives happen to every intrusion detection and intrusion prevention system.

This work proposes a mechanism for false positive/negative assessment with multiple IDSs/IPSs to collect FP and FN cases from real-world traffic and statistically analyze these cases. Over a period of 16 months, more than2000 FPs and FNs have been collected and analyzed. From the statistical analysis results, we obtain three interesting findings. First, more than 92.85 percent of false cases are FPs even if the numbers of attack types for FP and FN are similar. That is mainly because the behavior of applications or the format of the application content is self-defined; that is, there is not complete conformance to the specifications of RFCs. Accordingly, when this application meets an IDS/IPS with strict detection rules, its traffic will be regarded as malicious traffic, resulting in a lot of FPs. Second, about 91 percent of FP alerts, equal to about 85 percent of false cases, are no treated to security issues, but to management policy. For example, some companies and campuses limit or forbid their employees and students from using peer-to-peer applications; therefore, in order to easily detect P2P traffic, an IDS/IPS is configured to be sensitive to it. Hence, this causes alerts to be triggered easily regardless of whether the P2P application has malicious traffic or not. The last finding shows that buffer overflow, SQL server attacks, and worm slammer attacks account for 93 percent of FNs, even though they are aged attacks. This indicates that these attacks always have new variations to evade IDS/IPS detection.

2. LITERATURE SURVEY

In the area of network traffic classification, the state-of-the-art methods employ flow statistical features and machine learning techniques. Many supervised classification algorithms and unsupervised clustering algorithms have been applied to categorize Internet traffic. In supervised traffic classification, the traffic classes are predefined according to real applications and a set of labeled training samples are also manually collected for classifier construction. In contrast, the clustering-based methods can automatically group a set of unlabeled training samples and use the clustering results to train a traffic classifier. However, the number of clusters has to be set large enough to obtain useful and accurate traffic clusters, which results in a problem of mapping from a large number of traffic clusters to a small number of real applications. This problem is very difficult to solve without knowing any information about real applications.

In early works, Moore and Zuev applied the naïve Bayes techniques to classify network traffic based on the flow statistical features. Later, several well-known algorithms were also applied to traffic classification, such as Bayesian neural networks and support vector machines. Taking into account the real-time purpose, several supervised classification methods were proposed, which only used the first few packets. Other existing works include the Pearson's chi-Square test based technique, probability density function (PDF) based protocol fingerprints, and small time-windows based packet count. Different methods may have their own advantages in different network situations.

Some empirical study evaluated the traffic classification performance of different methods for practical usage. Roughan *et al.* have tested NN and LDA methods for traffic classification using five categories of statistical features. Williams *et al.* compared the supervised algorithms including naive Bayes with discretization, naive Bayes with kernel density estimation, C4.5 decision tree, Bayesian network and naive Bayes tree. Kim *et al.* extensively evaluated ports-based CorelReef method, host behavior-based BLINC method and seven common statistical feature based methods using supervised algorithms on seven different traffic traces. A recent research finding is that feature discretization is critical and essential for Internet traffic classification. By investigating the reasons for C4.5 performing very well under any circumstances, Lim *et al.* discovered that feature discretization can substantially improve the classification accuracy of every tested machine learning algorithm.

The performance of supervised methods is sensitive to the size of training data. Erman *et al.* proposed to use a set of supervised training data in an unsupervised approach to address the problem of mapping from flow clusters to real applications. However, the mapping method will produce a large proportion of 'unknown' clusters, especially when the supervised training data is very small. Another recent research finding is that flow correlation can be beneficial to traffic classification. Ma *et al.* proposed a payload-based clustering method for protocol inference, in which they grouped flows into equivalence clusters using a 3-tuple heuristic, i.e., the flows sharing the same destination IP, destination port and transport layer protocol are generated by the same application. Canini *et al.* tested the correctness of the 3-tuple heuristic with real-world traces. In our previous work, we applied the heuristic to improve unsupervised traffic clustering. However, it is unclear why flow correlation is helpful to traffic classification and how to apply flow correlation in the supervised classification approach. The problem of how to effectively classify network traffic using a small set of training data, is still to be solved.

3. EXISTING SYSTEM AND PROPOSED SYSTEM

3.1 Existing system

The Monitoring and analyzing of communication protocol in a connected device (a user/PC or system) is done using protocol based system. An application protocol consists of a system or agent that would typically sit within a group of servers, monitoring and analyzing the communication on application specific protocols. A host-based intrusion detection system (HIDS) consists of an agent on a host which identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability/acl databases) and other host activities and state. An example of a HIDS is OSSEC.

3.1.1 Disadvantages

1. Hackers recover the embedding data in original image as the data placed in particular bit position.
2. To attack the hidden data using original image because referred the key value.
3. The data extraction is not separable from the content descriptions.

3.2 Proposed System

The IDSs/IPSs can identify a normal activity as malicious one, causing a false positive (FP), or malicious traffic as normal, causing a false negative (FN) and then a variety of commercial products, open source, and research into IDSs were proposed and to create a pool of traffic traces causing possible FPs and FNs to IDSs because using Attack System Extraction (ASE). When securing a network, administrators have to use many different tools. Although functionality of them is similar, administrators have to spend a considerable amount of time to read documentation and learn how to use a new tool.

3.2.1 Advantages

1. Network Intrusion Detection Systems gain access to network traffic by connecting to a hub, network switch configured for port mirroring, or network tap.
2. To minimize this effort a specialized tool securing network and checking available service.
3. For each operating system different applications has to be used, regardless they are doing exactly the same.
4. The ASE was expanded into a bigger system, called the PCAPLib system. The PCAPLib system not only extracted and classified the real-world traffic captured from Campus Beta Site into proper categories by leveraging multiple IDSs, but also anonymized users privacy in these FP and FN traffic traces out of security considerations.

4. IMPLEMENTATION

4.1 Introduction of Technologies Used

Scoring packets: **Log-Version CLP:** To make it more suitable for real-time processing, conversion of floating-point division/multiplication operations into subtraction/addition operations is made. Scoring a packet is equivalent to looking up the scorebooks, e.g., the TTL scorebook, the packet size scorebook, the protocol type scorebook, etc. After looking up the multiple scorebooks, the matching CLP entries in a log-version scorebook are added. This is generally faster than multiplying the matching entries in a regular scorebook. This speed improvement becomes more beneficial as the number of scorebooks increases. On the other hand, generating a log-version scorebook may take longer than a regular scorebook generation. However, the scorebook is generated only once at the e end of each period and it is not necessary to observe every packet for scorebook generation; thus, some processing delay can be allowed. Furthermore, scorebook generation can be easily parallelized using two processing lines, which allows complete sampling without missing a packet.

Selective Packet Discarding: Once the score is computed for a packet, selective packet discarding, and overload control is performed using the score as the differentiating metric. Since an exact prioritization would require offline multiple-pass operations, e.g., sorting and packet buffering, the alternative approach is taken into account. First, the cumulative distribution function (CDF) of the scores of all incoming packets in time period (Ti) is maintained. Second, the cut-off threshold score is calculated. Third, the arriving packets in time period T (i+1) if its score value is below the cut-off threshold are discarded. At the same time, the packets arriving at T (i+1) create a new CDF.

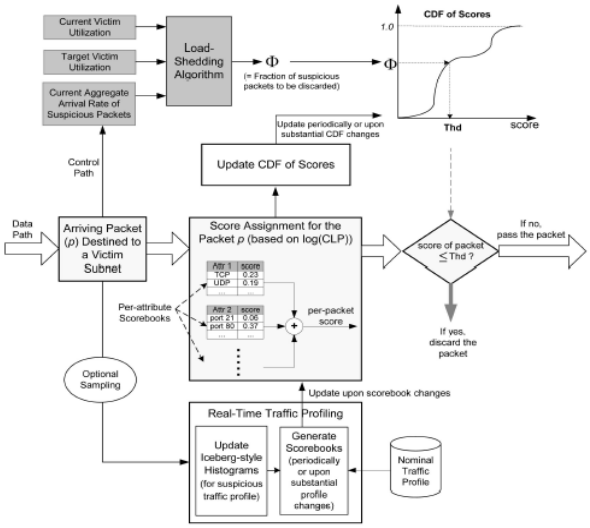

Fig 1: Packet differentiation and overload control.

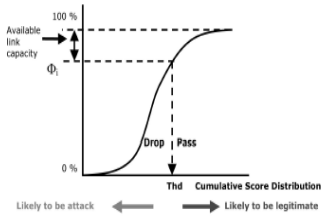

Fig 2: Selective packet discarding

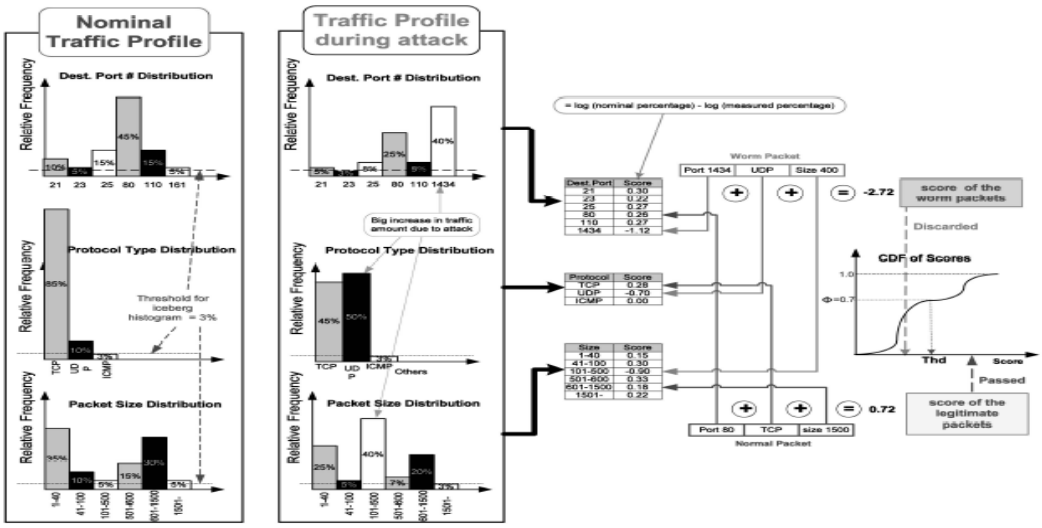

Fig 3: Discarding SQL Slammer Worm attack packets

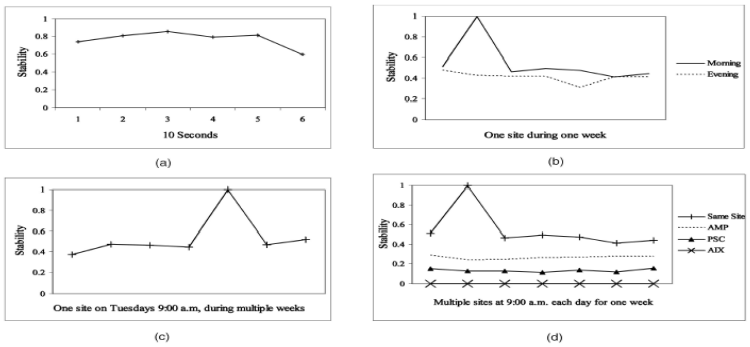

Fig 4: Traffic profile stability comparisons.

(a) Consecutive 10-second windows. (b) Seven consecutive days in a week. (c) Seven consecutive Tuesdays. (d) Four different sites for seven consecutive days in a week.

Packet Filtering Method – Algorithm

Step 1: Start the process.
Step 2: Trace the incoming packets using jpcap (java component used to capture the packets)
Step 3: Assign scores to the packets based on the Log Conditional LegitimatenProbability (CLP).
Step 4: Fix the threshold value which depends on the parameters like protocol type, packet size, etc.
Step 5: Compare the threshold with the packet score.
Step 6: If the packet score is greater than the threshold value then discard the packets.
Step 7: Update the cumulative distribution function score and continue with the Step 3.
Step 8: Stop the process.

Performance under Different Attack Types: The results are described in Table 5. In most cases, RA and RL were above 99 percent and false positives were kept low. The result was substantially better than random packet dropping of which the false positive ratio is expected to be 90.7 percent, and better than some of previous methods. Furthermore, *p*out was successfully kept close to its target value in most cases. The false negative ratios were mainly due to the gap between *p*target and *p*legitimate, i.e., the extra capacity left by the legitimate packets that allowed some attack packets to slip through. It is noteworthy that the false positive probability for the TCP-SYN flood attack was kept at a very low level (0 percent).
Although the signature of the TCP-SYN flood packets can easily be derived by any filtering scheme, the ability of PacketScore to prioritize legitimate TCP-SYN packets over attack packets based on other packet attributes is an essential feature. Without such prioritization, e.g., in the case of stateless rule/signature-based filtering, all TCPSYN packets would be discarded and, thus, ensure the success of the DDoS attack on the victim. PacketScore did show some degradation under nominal attack when the TTL values were randomized. Legitimate packets having the same characteristics as attack packets were penalized, but such chances were still quite small, and the false positive ratio was kept to 3.30 percent. When the TTL values were fixed, the performance was better. The TTL value 118 accounted for the largest portion of traffic (29.86 percent) and 100 had a ratio under the adaptive threshold (0.29 percent) for 99 percent coverage. As PacketScore utilizes more attributes, the performance should become better. Only one joint attribute was used in this experiment, numerous combinations of attributes are possible to increase the number of

attributes, and the performance under nominal attack can be further improved. It is interesting to see how the scores of each attack type are distributed.

TABLE 1: Performance against Different Types of Attack

Attack Type	% False positive	% False negative	PDF Separation		$\frac{\rho_{mt}}{\rho_{target}}$
			% R_A	% R_L	
Generic	0.04	4.03	99.9	99.9	0.99
SYN flood	0.0	4.03	100.0	100.0	0.99
SQL Worm	0.0	4.06	100.0	100.0	0.99
Nominal (TTL=100)	0.0	2.68	100.0	100.0	0.85
Nominal (TTL=118)	0.09	3.25	100.0	99.9	0.91
Nominal (TTL=random)	3.30	4.24	93.6	94.7	0.99
Mixed (TTL=100 for nominal)	0.01	4.42	99.8	99.7	1.14

One challenging issue of the PacketScore scheme is the need for a clean baseline profile as in other profile-based systems. This is because DDoS attack traffic is already prevalent on the Internet and a quiet attack-free period may be hard to find. As a result, the constructed nominal profile may be biased by the DDoS traffic and may force PacketScore to accept DDoS traffic that has been reflected in the profile. However, PacketScore is designed to accept specific traffic only up to the maximum ratio that was observed in the past. Therefore, DDoS traffic beyond this ratio will be properly filtered by PacketScore and, thus, cannot succeed in a massive attack. Nevertheless, accepting DDoS traffic is not desirable as it wastes bandwidth. This situation can be improved by constructing a cleaner nominal profile that contains less DDoS traffic. A cleaner profile can be made one of two ways. First, the packet trace data can be analyzed to identify legitimate flows that show proper two-way communication behavior. The packets from the legitimate flows are used for constructing the profile. Although some traffic flows that do not have continuous packet exchange, such as ICMP, may be left out, PacketScore filtering is already based on an iceberg-style histogram with default value assignment for noniceberg items. The impact on performance of missing some of the packets should be minimal. Second, a generic profile to remove those packets that are more likely to be attack packets can be used first, and use the remaining packets to create the final nominal profile. The generic profile reflects overall Internet traffic characteristics, e.g., TCP versus UDP ratio, common packet size, common TCP flags, etc. Our preliminary research shows that this two-step profiling is very effective to fight generic attacks. Further study is needed on these methods.

Traffic Classification: Traffic classification is an automated process which categories computer network traffic according to various parameters (for example, based on port number or protocol) into a number of traffic classes. Each resulting traffic class can be treated differently to differentiate the service implied for the user (data generator/ consumer).

Typical uses: Once packets have been classified, for example, each traffic class could be subject to a different rate limit, shaped separately and/or prioritized relative to other traffic classes. This differentiation can be used by a network operator to treat different types of application traffic differently (for example, prioritize voice over file sharing for the responsiveness perceived by end users), and to offer premium services at a higher price point than basic ones. Traffic classification is a cornerstone of the differentiated treatment of Internet traffic, including some data discrimination techniques, and consequently is an important and at times controversial factor in debates on network neutrality. Traffic classification is one of several mechanisms used in teletraffic engineering and traffic management in IP and ATMnetworks. Differentiated Services specifies that packets are marked according to their class, determined by a traffic classifier - a node in the network which assesses which class a particular packet should belong to, and marks it with a Differentiated Services Code Point (or DSCP) accordingly.

Implementation: Classification is achieved by various means. Matching bit patterns of data to those of known protocols is a simple, yet widely-used technique. An example to match the protocol handshaking phase would be a check to see if a packet began with character 19 which was then followed by the 19-byte string 'BitTorrent protocol'. More advanced traffic classification techniques rely on statistical analysis of attributes such as byte frequencies, packet sizes and packet inter-arrival times. Upon classifying a traffic flow using a particular protocol, a predetermined policy can be applied to it and other flows to either guarantee a certain quality (as with VoIP or media streaming service[1]) or to provide best-effort delivery. This may be applied at the ingress point (the point at which traffic enters the network) with a granularity that allows traffic management mechanisms to separate traffic into individual flows and queue, police and shape them differently.

Typical traffic classes: Operators often distinguish three broad types of network traffic:

Sensitive traffic: Sensitive traffic is traffic the operator has an expectation to deliver on time. This includes VoIP, online gaming, video conferencing, and web browsing. Traffic management schemes are typically tailored in such a way that the quality of service of these selected uses is guaranteed, or at least prioritized over other classes of traffic. This can be accomplished by the absence of shaping for this traffic class, or by prioritizing sensitive traffic above other classes.

Best-effort traffic: Best effort traffic is all other kinds of non-detrimental traffic. This is traffic that the ISP deems isn't sensitive to Quality of Service metrics (jitter, packet loss, latency). A typical example would be peer-to-peer and email applications. Traffic management schemes are generally tailored so best-effort traffic gets what is left after sensitive traffic.

Undesired traffic: This category is generally limited to the delivery of spam and traffic created by worms, botnets, and other malicious attacks. In some networks, this definition can include such traffic as non-local VoIP (for example, Skype) or video streaming services to protect the market for the 'in-house' services of the same type. In these cases, traffic classification mechanisms identify this traffic, allowing the network operator to either block this traffic entirely, or severely hamper its operation.

5. RESULTS AND DISCUSSION

The following steps and snapshots show the actual working scenario and outcomes of this system.

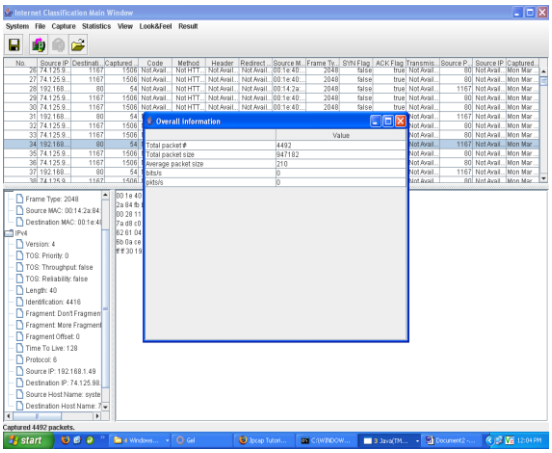

Fig. 1: Overall information of the traffic flow

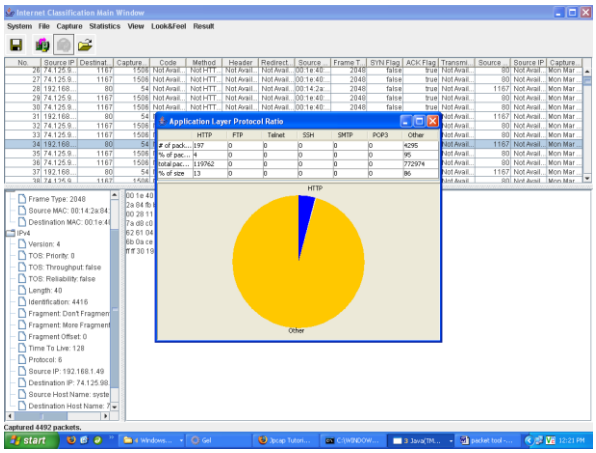

Fig. 2: Application layer protocol ratio

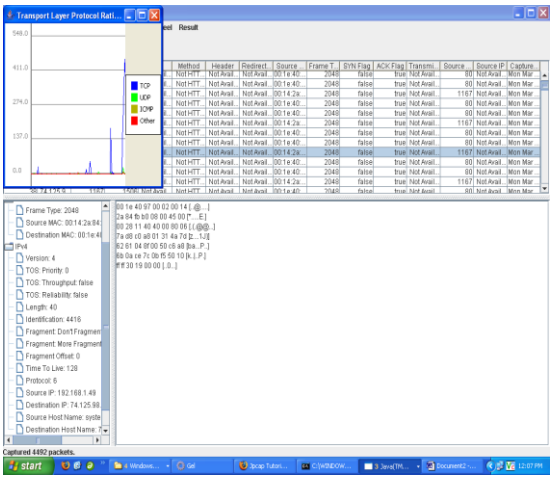

Fig. 3: Transport layer protocol ratio

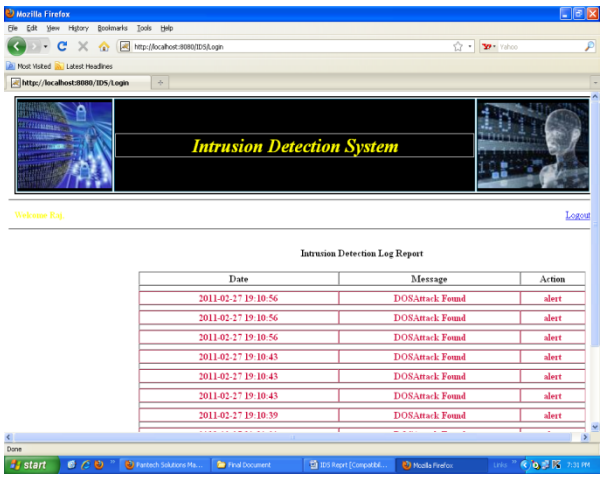

Fig. 4: Intrusion detection result

6. CONCLUSION

Packet analysis has been shown as an approach that improves the state of the art by generating packet filters that combine most of the desired properties in terms of processing speed, memory consumption, flexibility and simplicity in specifying protocol formats and filtering rules, effective filter composition, and low run-time overhead for safety enforcement. The development of the filter generator and the test results support the viability of our claims. We have designed, prototyped, and evaluated SPAF, a packet filter generator based on the creation of finite-state automata from a high-level protocol format database and filter predicates. It aims at emitting fast and efficient filters while preserving all the relevant safety properties, both in terms of memory access correctness and termination.

7. FUTURE WORK

Since the size of the network traffic collected is huge, there is a need to store them in a specific database, so later our tool can query the data directly from the server. We are Building an interface between the tool and the database server using JDBC connection. We also plan to store the result back to the database server once the user decided to save the result. The other feature we are currently working is to add pattern recognition algorithms for intrusion detection purpose since the tool is able to pre-

process the network connection data. It would be useful to add another module to this tool so that it can perform some preliminary analysis for network intrusion detection such as clustering similar network connections according to similar patterns e.g., payload or network protocols.

References

[1]. K.-C. Lan, A. Hussain, and D. Dutta, "Effect of Malicious Traffic on The Network," Proc. Passive and Active Measurement Wksp. (PAM), San Diego, CA, Apr. 2003.

[2]. S.-X. Wu and W. Banzhaf, "The Use of Computational Intelligence in Intrusion Detection Systems: A Review," Detection," Proc. 16th Int'l. Conf. Systems, Signals and Image Processing, June 2009.

[3]. I.-W. Chen et al., "Extracting Attack Sessions from Real Traffic with Intrusion Prevention Systems," Proc. IEEE ICC, June 2009.

[4]. S.-H. Wang, "Extracting, Classifying and Anonym zing Packet Traces with Case Studies on False Positives/Negatives Assessment," M.S. thesis, Dept. Comp. Sci., Nat'l. Chiao Tung Univ., Taiwan, 2010.

[5]. Y.-D. Lin et al., "On Campus Beta Site: Architecture Designs, Operational Experience, and Top Product Defects," IEEE Commun. Mag., vol. 48, no. 12, Dec. 2010, pp. 83–91.