IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 3, Issue 2, April-May, 2015
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

Enhancement In Real Time Network Traffic Classifier Using Packet Capture Library

R. Parimalam[1], Prof. P. Krishna Kumar[2]
Department of Computer Science Engineering, PET Engineering College, Vallioor, India

***Abstract*— The existing PCAPLib system extract, classify, and anonymize the packet traces from real network traffic with two mechanisms. Initially, active trace collection (ATC) actively extracts and classifies packet traces into sessions by leveraging multiple detection devices. Second, deep packet anonymization (DPA) protects the privacy in the packet payloads for hundreds of application protocols while preserving the utility of the traces. In the existing technique, there is an issue occurring in packet capture size, duration, average packet size and average packet rate is not identified. This work proposes an enhancement of real time network traffic classifier using packet capture library. In addition, the deep packet tracing is used to trace the packet with packet capture size and average packet size with packet capture duration and average packet rate. Enhanced packet capture system automatically extract and classify application sessions from bulk traffic in a scalable manner. The packet traces can be flexibly anonymized for protecting privacy and used for network analysis of various purposes not just for network security study. The PCAPLib system has been operating for a long time to extract the application sessions continuously. The enhanced packet traces provided to cooperating organizations and are planned to be made public in the future. We are also work toward the improvement of packet anonymization and it will be effectively integrated with the advance packet capture library and ease locating sensitive application fields to be anonymized.**

Keywords— *Packet capturing library, Active trace collection, Deep packet anonymization.*

1. INTRODUCTION

The Internet traffic is changing continuously and this contribute to difficult the characterization of network behavior and structure. Massive games and cloud and grid services increase every day their percentage participation in total network traffic (1). Due to the rapid growth of the Internet and the multitude of new services and protocols, there have been reported large numbers of cases of misuse and attacks on those services and more continue being reported each year (2, 3). During the last several years, malicious traffic detection has been an active area of network security because the Internet has witnessed a surge in malicious traffic generated by network attacks, such as denial-of-service (DoS) and propagation of botnets, viruses, worms, trojan horses, spyware and so on. Moreover, malicious traffic makes network performance inefficient and troubles users (4). For example, distributed DoS (DDoS) attacks increase Domain Name Service (DNS) latencies by 230 percent and web latencies by 30 percent. During July-August 2001, 395,000 computers were infected worldwide with the CodeRed worm, which resulted in approximately $2.6 billion in damages (5).

There are a multitude of malicious traffic detection techniques and thus vulnerabilities in common security components, such as firewalls, are unavoidable. Today, Intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) are commonly used to detect different types of malicious traffic, network communications and computer system usage with the mission of preserving systems from widespread damage; that is because other detection and prevention techniques, such as firewalls, access control, skepticism, and encryption, have failed to fully protect networks and computer systems from increasingly sophisticated attacks and malware (4, 6, 7).

Real world Internet traffic is useful for studies ranging from traffic characterization and analysis, diagnosis of network events, to evaluation of network systems such as intrusion detection/prevention (IDP) systems. Network research communities, as well as developers of network appliances, usually rely on large, diverse, updated and non synthetic packet traces for experimental study and evaluation. For example, network analysts who collaborate on inspecting malicious incidents and network behavior can share traces among one another. Capturing and sharing real traffic face two major challenges. First, the packet traces in a large repository involve diverse application protocols and they should be well classified beforehand for users to easily find the desirable traces. Several organizations have released packet traces for public access, e.g., (8, 9). The traces are usually submitted and categorized subjectively by enthusiastic contributors. They are often outdated, inconsistently categorized and even unusable, as the packet repository lacks a central control mechanism to maintain the quality of packet traces and consistently categorize them. Moreover, if the packet traces are derived from a real environment, e.g., from an Internet service provider, the volume will be usually huge and it is essential to automatically extract and consistently classify the packet traces in a scalable way. Second, the packet traces may contain private information such as host addresses, e-mail addresses, and even authentication keys. Such information should be anonymized before the traces are shared. Although packet anonymization helps to protect the privacy of packet traces, it hurts their utility at the same time. Many existing methods anonymize only the fields in the Transmission Control Protocol/protocol suite (TCP/IP) header and strip away the payloads. However, the payloads are likely to contain key information for network analysis in terms of signature matching for intrusions, traffic identification or payload-based anomaly detection. Several research works have attempted to preserve the payloads and anonymize specified key information in them. The anonymization methods are still limited to only a few common protocols (e.g., HTTP and FTP) and unable to satisfy the need for large packet traces, which may be from a large number of application protocols. Although it is possible to extend existing works to support the anonymization for so many application protocols, the support will require implementing that many protocol parsers as well, yet the implementation effort is nontrivial (10).

2. BACKGROUND AND RELATED WORK

Packet capturing anonymization (PCAPAnon) provides hundreds of protocol parsers based on Wire shark dissectors which can be applied to all packet fields up to the application level. So many application protocols have been ready to use with the support of anonymizing and protocol parsing allows precise specification of the right fields for anonymization. To preserve the statistical individuality of packets as much as possible, PCAPAnon replaces the field values with those of the same semantics (e.g., replacing a URL with another URL) and length in the anonymization. This defense will benefit the methods that rely on statistical characteristics for traffic analysis and it also prevents the protocol parsers from triggering an error during the parsing process since the semantics of application fields remain the same.

The Active Trace Collection (ATC) device can automatically classify and extract application sessions from bulk network traffic in a real environment such as Beta Site and it is essential to continue the repository of packet traces in a scalable manner. A ready-to-use implementation of the PCAPAnon mechanism can anonymize packet traces for hundreds of application protocols in existing practices though preserving both the semantics and length of important application fields to keep the utility for later analysis. The value of this system for network analysis has been demonstrated by IDP systems. It is noted that the packet traces are not only useful for intrusion detection but also for various kinds of network traffic analysis. The implementation of this system is publicly available as a Source Forge project. The packet traces in the assessment are also available. The disadvantages of the existing systems include that

* The data set was criticized for being too old to reflect contemporary network traffic.
* The packet traces in a large repository involve diverse application protocols.
* Many existing methods anonymized only the fields in the Transmission Control Protocol.
* The anonymization methods are still limited to only a few common protocols.

3. PROPOSED SYSTEM

The effectiveness of machine learning techniques were evaluated for the real time traffic classification problem using statistical attributes derived from first packets of each flow. Port-base method was utilized for labeling flows into categories. The acquired results showed that the classification with decision trees had the highest accuracy and performance in comparison of other classifiers. In addition, sub flow-based classifiers can reach high accuracy values while the computational complexity is reduced. PCAPAnon wires the configuration to facilitate the users to specify the fields in hundreds of application protocols to be present anonymized with consistent alteration while the reliability and utility of the packet traces were preserved. It also compares PCAPLib among the recent ISCX information sets and is available upon requests.

It claims to be free from privacy issues because the data set are emulated from a controlled tested environment based on the profiles describing the do violence to scenario and the statistical distribution from real networks and also consider the comparison with binary protocols. The comparison is harder than that with the preceding for human readable ASCII protocols because neither standard nor anon tool has the parsers of the binary protocols and it is difficult to specify sensitive patterns for binary protocol in anon tool except for a few cases such as the area names appearing in the DNS queries. Even an IP address in a DNS response cannot be specified because a pattern because it is represented in a 4-byte binary value. If the payloads are reduced by default, the utility will be seriously affected since the DUT relies on deep packet inspection for malicious signatures in the packet payloads. Advantages of the proposed system includes

* Privacy, Utility, and Efficiency
* PCAPAnon uses customized parsing to hide more identities than anon tool.
* To avoid missing sensitive application fields due to careless specification.
* PCAPAnon also supports pattern substitution with regular expression

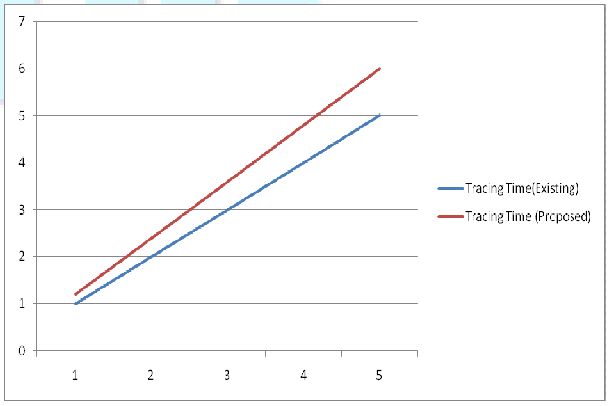

Fig. 1: Comparison with existing system

4.SYSTEM ARCHITECTURE

The system architecture is shown in figure 2.

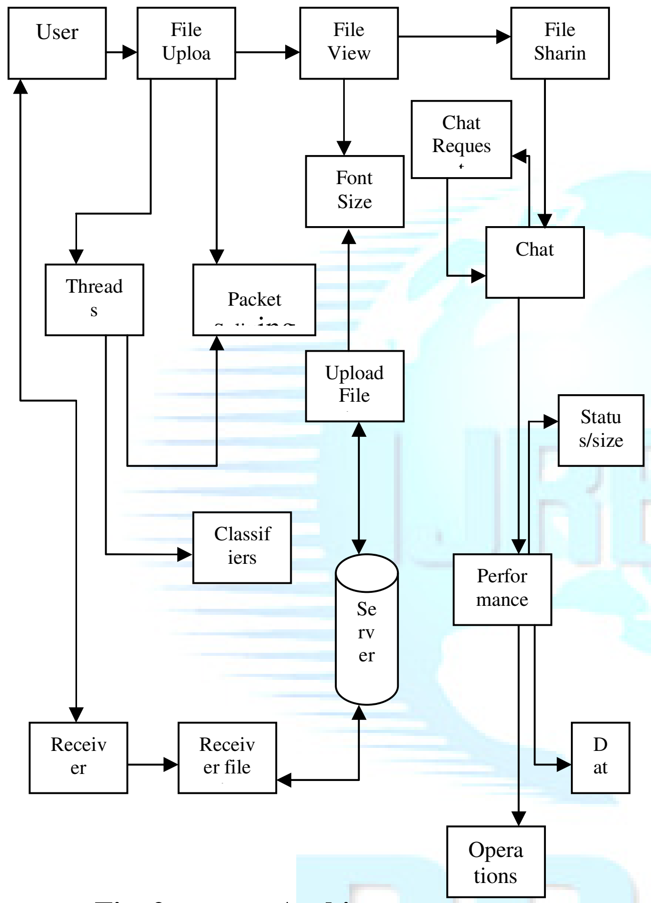

Fig. 2: system Architecture

5. SYSTEM IMPLEMENTATION

5.1 Real Packet Traces

Emulated packet traces can be generated in the laboratory by custom scripts or traffic generators such as Harpoon. A well-known example of this approach is the DARPA-sponsored evaluation data set for intrusion detection but the data set was criticized for being too old to reflect contemporary network traffic. A scalable method to automatically extract and classify the packet traces is therefore required. Moreover, the network operators may be unwilling to allow the researchers to acquire the traces due to privacy concerns.

5.2Active Trace Collection

In contrast, the ATC mechanism can automatically capture, extract and classify large scale packet traces from real traffic. A traffic replay tool replays captured raw traffic to multiple devices under test (DUTs) to leverage their domain knowledge. The ATC associates the sessions according to the log messages and classifies the traces into different categories by matching the logs with representative keywords.

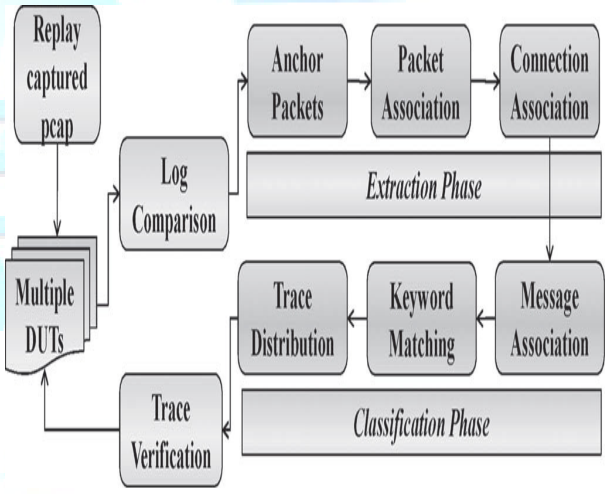

Fig. 3: ATC system flow

5.2 PCAPAnon

PCAPAnon provides hundreds of protocol parsers based on wire shark dissectors which can be applied to all packet fields up to the application level. Support of anonymizings, so many application protocols has been ready to use and protocol parsing allow precise specification of the right field for anonymization. To preserve the statistical characteristics of packets as much as possible, PCAPAnon replaces the field values with those of the same semantics (e.g., replace a URL with another URL) and length in the anonymization. The preservation determination benefit the methods that rely on statistical individuality for traffic analysis and it also prevents the protocol parsers from triggering an error during the parsing process since the semantics of application fields remain the same.

5.3 Trace Repository

The Wire shark code is modified to support anonymizing the specified application fields from the command options or custom scripts based on subjective judgment on sensitive fields or analysis from the heuristics for the options mean anonym force the content in the HTTP HOST field. The specification is particularly practical when an application field is private (e.g., a password) yet difficult to specify the content with a pattern. To avoid missing sensitive application fields due to careless specification, PCAPAnon also wires pattern substitution with regular expression (Rage) matching to seek and match sensitive identities such as IP addresses, mail addresses and URLs. The substitution is also useful if the identities are from an unknown application protocol or not precisely identified in protocol parsing.

5.4 Packet splitting as threads

In this module the uploaded file will be splits as packets for the packet split and uploaded. These split process will follows as file process. In this process how many packets wants to split the packets. The split packets are also known as privacy and integrity threads. The main purpose of this module is maintains privacy.

5.5 Performance of monitoring process

Enhancement of real time network traffic classifier using packet capture library is proposed. In addition, the deep packet tracing is used to trace the packet with packet capture size and average packet size with packet capture duration and average packet rate.

6. SYSTEM TESTING

Testing is an important phase encountered in any developed product or framework. Because, the developed product should be free from errors and it should be validated for accuracy. The product should work under normal conditions as long as the user gives proper inputs and therefore it should be checked for its robustness and should withstand and inform the users about the erroneous input. The testing phase involves testing the system using various test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system is tested using those test data. Errors found are corrected and recorded for future reference. Thus a series of testing is performed on the system before it is ready for implementation. Testing is applied at the different levels of development cycle. Each level of testing is different in nature and has different objectives at each level. The focus of all levels of testing is to find errors, but different types of errors are looked for at each level. The quality of system is confirmed by the thoroughness of its testing. Duration and cost of testing and debugging is a significant fraction of the system development cycle and hence influences overall productivity during the development. In this phase, the errors in the programs or modules are localized and modifications are done to eliminate them. The testing makes a logical assumption that all parts of the system work efficiently and the goal is achieved.

Black-box testing is an approach to testing where the tests are derived from the program or component specification. The system is a 'block box' whose behavior can only be determined by studying its inputs and the related outputs. It is also known as functional testing because the tester is only concerned with the functionality and not the implementation of the software. The methods commonly used here are Equivalence partitioning, Boundary-value analysis and Error guessing.

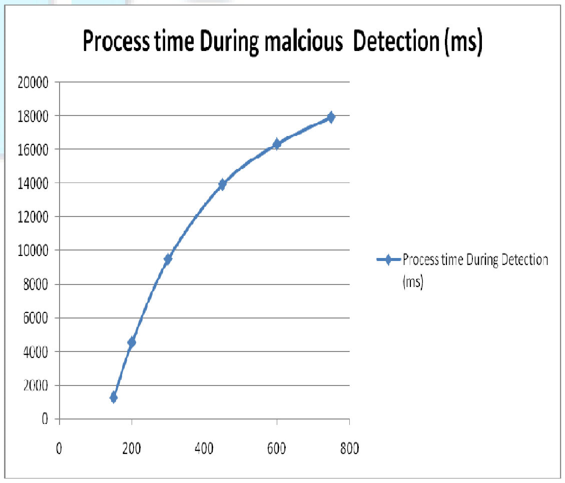

Fig. 4:Performance evaluation

Equivalence partitioning is a systematic process that identifies on the basis of whatever information is available, a set of interesting classes of input conditions to be tested, where each class is representative of a large set of other possible tests.

In boundary value analysis, the elements are selected such that each edge of the EC is the subject of a test. Example: If an input specifies a range of valid values, write test cases for the ends of the range and invalid-input test cases for conditions just beyond the ends. If the input requires a real number in the range 0.0 and 90.0 degrees, then write test cases for 0.0,90.0.

Error guessing is an ad hoc approach; identify the tests that are considered likely to expose errors. The basic idea is to make a list of possible errors or error-prone situations and then develop tests based on the list.

1) Low level testing involves testing individual program components one at a time or in combination. It requires intimate knowledge of the program's internal structure. The objective of this testing and integration testing is that the code implemented the design properly. In developing a large system testing usually involves several stages. These test cases are more refined and are generally written with details such as 'Expected Result', 'Test Data', etc.

Unit testing focuses on the verification effort of the smallest unit of design module. Attention is diverted to individual modules, independently to locate errors. This has enabled the detection of errors in coding and logic. The various modules of the system are tested in unit testing method. Using the detailed description as a guide, important control parts are tested to uncover errors within the boundary of the module. The relative complexity of tests and the error detected as a result is limited by the constrained scope established for unit testing. This test focuses on each module individually, ensuring that it functions properly as a unit, and hence the name Unit Testing.

Module tests seek to validate the code produced to create sets of logically connected subroutines and data which have been grouping together into modules. Module testing is concerned with testing the smallest piece of software for which a separate specification exists. After checking for errors the modules can be integrated.

Integration testing is carried out after the modules are integrated. This test uncovers the errors associated with the interface. This testing is done with sample data. The need for integration is to find overall system performance. The objective is to take unit tested modules to build a programmed structure. All the modules are combined and tested as whole. Thus in integration testing step, all the errors uncovered are corrected for the next testing steps.

7. CONCLUSION AND FUTURE WORK:

Architecture, implementation, and performance of an Internet traffic classifier monitor were presented in this work. The monitor is composed of three modules which were implemented as concurrent processes: capture and pre-processing, flow reassembly and classification. For the traffic trace, the throughput reassembly module of the current implementation is 24997.25 flows per second. The system has been proven effective by running it in an operational network, the Beta Site, and its source code is presented as a Source Forge project at the implementation of this system is publicly accessible as a Source Forge project the packet traces in the evaluation also exist in. It presents the background and connected work describes the plan and ideas of our methodology addresses the major system execution issues.

Future directions for this research includes incorporating sub flow based classification in ITCM to reduce response time. Second, we aim to verify the performance impact of our classifier monitor at gigabit links, which are becoming increasingly common at computer networks. Finally, we could also prototype ITCM with NetFPGA hardware, since the implementation of network systems in hardware is essential for any real time application, particularly in gigabit networks.

REFERENCES

1. Silas Santiago Lopes Pereira, Jose Everardo Bessa Maia, Jorge Luiz de Castro e Silva. ITCM: A real time internet traffic classifier

monitor. International Journal of Computer Science & Information Technology 2014; 6 (6): 23 – 38. DOI:10.5121/ijcsit.2014.6602

2. Sans top 20 security risks for 2007. http://www.sans.org/top20/

3. Michael Foukarakis, Demetres Antoniades, Michalis Polychronakis. Deep Packet Anonymization. EUROSEC 2009. Proceedings of the Second European Workshop on System Security. 16 - 21. Doi: 10.1145/1519144.1519147

4. Cheng-Yuan Ho, Ying-Dar Lin, Yuan-Cheng Lai, I-Wei Chen, Fu-Yu Wang, Wei-Hsuan Tai. False Positives and Negatives from Real Traffic with Intrusion Detection/Prevention Systems. International Journal of Future Computer and Communication 2012; 1(2): 87 – 90.

5. Cheng-Yuan Ho, Yuan-Cheng Lai, I-Wei Chen, Fu-Yu Wang, Wei-Hsuan Tai. Statistical Analysis of False Positives and False Negatives from Real Traffic with Intrusion Detection/Prevention Systems. IEEE Communication Magazine 2012; 50 (3): 146 – 154. Doi: 10.1109/MCOM.2012.6163595

6. S. X. Wu, W. Banzhaf. The Use of Computational Intelligence in IntrusionDetection Systems: A Review. Elsevier Applied Soft Computing 2010; 10: 1 – 35.

7. H. T. Elshoush, I. M. Osman. Reducing False Positives through Fuzzy Alert Correlation in Collaborative Intelligent Intrusion Detection Systems — A Review. Proc. of IEEE International Conference on Fuzzy Systems (FUZZ), July 2000; 1 – 8.

8. PCAPR collaborative network forensics. [Online]. Available: http://www.pcapr.net/forensics

9. Packetlife repository. [Online]. Available: http://www.packetlife.net/ captures

10. Ying-Dar Lin, Po-Ching Lin, Sheng-Hao Wang, I-Wei Chen, Yuan-Cheng Lai. PCAPLib: A System of Extracting, Classifying and Anonymizing Real Packet Traces. Systems Journal, IEEE 2014; PP (99): 1 – 12. Doi: 10.1109/JSYST.2014.2301464