International Journal of Engineering Inventions
ISSN: 2278-7461, www.ijeijournal.com
Volume 1, Issue 2 (September 2012) PP: 40-46

Performance Improvement of PPM Algorithm

Pramila B [1], Arunkumar G [2], Jagadisha N[3], Prasad M.R[4]
[1]*Department of E&CE, EWIT, Bangalore.*
[2]*Department of E&CE, STJIT, Ranebennur.*
[3]*Department of ISE, EWIT, Bangalore.*
[4]*Department of CSE, EWIT, Bangalore.*

Abstract—While Denial-of-Service (DoS) attack technology continues to evolve, the circumstances enabling attacks have not significantly changed in recent years. DoS attacks remain a serious threat to the users, organizations, and infrastructures of the Internet. The research work highlights the usage of probabilistic packet marking (PPM) algorithm which is a promising way to discover the Internet map or an attack path that the attack packets traversed during a distributed denial-of-service attack. But it has been seen that its termination condition is not well defined which results in incorrect attack path constructed by the PPM algorithm. In this research work, a novel approach for a precise termination condition named as rectified PPM (RPPM) algorithm. The most significant merit of the RPPM algorithm is that when the algorithm terminates, the algorithm guarantees that the constructed attack path is correct, with a specified level of confidence. An experimental framework is designed on the RPPM algorithm and show that the RPPM algorithm can guarantee the correctness of the constructed attack path under different probabilities that a router marks the attack packets and different structures.

Keywords—Denial-of-Service (DoS), Probabilistic Packet Marking (PPM), Termination Packet Number (TPN).

I. INTRODUCTION

Today, the Internet is an essential part of our everyday life and many important and crucial services like banking, shopping, transport, health, and communication are partly or completely dependent on the Internet. According to recent sources the number of hosts connected to the internet has increased to almost 400 million and there are currently more than 1 billion users of the Internet. Thus, any disruption in the operation of the Internet can be very inconvenient for most of us. The devastating effects of the DoS attack has caused attention of scientists and researchers, leading to various mechanisms that have been proposed to deal with them. However, most of them are ineffective against massively distributed DoS attacks involving thousands of compromised machines. IP trace back is an important step in defending against Denial-of-service (DoS) attacks. Probabilistic packet marking (PPM) has been studied as a promising approach to realize IP traceback.

The packet marking procedure is designed to randomly encode edges' information on the packets arriving at the routers. Then, by using the information, the victim executes the path reconstruction procedure to construct the attack path. Many IP traceback techniques have been proposed in the past. Among them the Probabilistic Packet Marking (PPM) approach has been studied mostly. In a PPM approach, the router probabilistically marks the packets with its identification information and then the destination reconstructs the network path by combining a number of such marked packets. In the existing system, PPM algorithm is not perfect, as its termination condition is not well defined. The algorithm requires prior knowledge about the network topology. In packet marking algorithm, the Termination Packet Number (TPN) calculation is not well defined in the literature. In the existing system it only supports the single attacker environment.

II. LITERATURE SURVEY

The denial-of-service (DoS) attack is a pressing problem for recent years. DoS defense research blossomed into one of the main stream in the computer security fields. Various techniques such as the pushback message, ICMP traceback, and the packet filtering techniques are the results from this active field of research.

Denial of service (DoS) or Distributed DoS (DDoS) attacks have become one of the most severe network attacks today. Though relatively easy to be executed [1], it could cause devastating damages. By consuming a huge amount of system resources, DoS attacks can render the normal services to the legitimate users unavailable. While email has become the most popular form of communication, the DDoS attack is a common mode of attack to cripple a mail server [2].

Lee and Fung [3] indicate that a DoS attack could be carried out during an authentication process involving public-key based operations. Many different approaches have been proposed to defend against DoS attacks. To mitigate the damage from DDoS attack, Su et al. [6] proposed an online approach which involves first identifying the approximate sources of the an attack traffic, and then applying packet filtering as near the attack sources as possible.

Huang et al. [7] proposed an incentive based approach using cooperative filtering and cooperative caching for defeating DDoS attacks. The most effective approach against DoS attack is to isolate the attackers from the victim's network. Thus, locating the attack source would be most important. We cannot rely on the source address in the *IP* header of an attack packet since the source address is not authenticated in the current protocol when a router forwards a packet; so the attacker can spoof the source *IP* address while launching an attack. Locating the attack source usually involves finding the

paths of the relevant packets. Because of the stateless nature of Internet routing, it is quite difficult to identify such paths. Finding the attack traffic paths is known as the IP traceback problem [9].

Dean et al. proposed an algebraic marking scheme for IP traceback ,which is based on linear algebra and coding theory. One main drawback of this marking scheme is its ineffectiveness in dealing with multiple attacks. DoS and Distributed Denial of Service (DDoS) attacks have posed major security threats to the Internet. To define, a DoS is a malicious attempt to render a node or network incapable of servicing the legitimate requests. DDoS is the case in which one or more attackers coordinate the sending of enormous packets aiming at clogging the victim and the network. DDoS attacks come from various sources with different types of attacks. These attacks attempt to consume the finite resources like memory, computational power etc. in the network and also at the victim [1]. They would also result in heavy congestion on the network links thereby disrupting the communication among the users. Moreover, these attacks would exploit the inherent weaknesses in the IP protocol. One such is spoofing, which is, impersonating one's IP address. Spoofing further complicates the detection of the DoS / DDoS attacks. More importantly, innocent host systems are involved for launching such attacks. For instance, the DoS attacks have disrupted the Internet services and incurred heavy financial losses [2]. Such attacks are difficult to detect, prevent and traceback, but easy to implement. Thus the devastating effects of the problem have led to the development of many anti-DoS solutions.

The countermeasures that address the DoS attacks are deployed at different points on the Internet namely at the source-end, intermediate-network and victim-end. An ideal place to detect and filter the flooding attacks is at the network where they have been generated that is as close to the source of attacks as possible. The source-end defenses act as filters for the attacks and have advantages over the other two [3]. They include congestion avoidance over the network, small collateral damage, feasible deployment etc. The source-end defensive systems maintain the statistics of the outgoing traffic and use them to analyze and conclude about the ongoing attacks. But, because of the highly distributed nature of the attacks, detection near the source is cumbersome. Attackers may try to launch the attacks that resemble legitimate requests, thus there is no suspicious alert about the attack traffic can be generated.

The Probabilistic Packet Marking (PPM) algorithm by Savage et al. had attracted the most attentions in contributing the idea of IP traceback. The most interesting point of this IP traceback approach is that it allows routers to mark certain information on attack packets based on a pre-determined probability. Upon receiving enough marked packets, the victim (or a data collection node) constructs an attack path, which is the set of paths the attack packets traversed Logging involves storing the packets on the routers which they come across and using data mining principles to find the path the packets traversed. This method can even trace back the attack long after it has been over. Since the process of logging incurs significant storage overhead on routers, a mechanism that stores packet digests rather than the packets themselves has been addressed by Snoren et. al. [4]. Although the hash based technique requires 0.5% of the total link capacity in digest table storage, the storage requirement at the routers, in general, would be too high.

III. DESIGN AND IMPLEMENTATION

Implementation of any software is always preceded by important decisions regarding selection of the platform, the language used, etc. These decisions are often influenced by several factors such as the real environment in which the system works, the speed that is required, the security concerns, other implementation specific details etc.,

There are three major implementation decisions that have been made before the implementation of this project. They are as follows

- Selection of the platform (operating system).
- Selection of the programming language for development of the system.
- Coding Standards.

A. Implementation Requirements

The implementation decisions are mainly concerned with the ease of future enhancement of the system. The major implementation requirements are

- Operating System used to implement this project is Windows XP.
- The language chosen for this project is Java
- Java Swing.

B. Path Selection

Windows is distinguished from many popular operating systems in the following important ways. The Windows operating system is designed to be compatible with the largest combination of PC hardware. It has vast compatibility with motherboards, processors, memory chips, USB devices, internal disk drives of the PATA and SATA variety and no doubt other devices and standards yet to be designed can all be configured to run under Windows. Windows desktop and its simple user interface have become synonymous with computing. Linux operating systems offer a similar mouse driven interface and the Apple Mac continues to pioneer with its single button mouse, but Windows remains the most popular and the most recognizable simply by virtue of its earlier successes and the work done by each release of the operating system to maintain this user familiarity.

C. Language selection

For the implementation of our project we need flexible systems implementation language. Compilation should be relatively straightforward compiler, provide low-level access to memory, provide language constructs that map efficiently to machine instructions, and require minimal run-time support. Program should be compiled for a very wide variety of computer platforms and operating systems with minimal change to its source code. For Graphical User Interface

programming, language chosen must be simple to uses, secure, architecture neutral and portable. Additional requirements of GUI are: 1) User interface management: Windows, menus, toolbars and other presentation components be supported by the language.2) Data and presentation management: language must contains a rich toolset for presenting data to the user and manipulating that data. 3) The Editor: The language should have an editor, a powerful and extensible toolset for building custom editors. 4) The Wizard framework: A toolset for easily building extensible, user-friendly Wizards to guide users through more complex tasks. 5) Configuration management: Rather than tediously write code to access remote data and manage and save user-configurable settings, etc., all of this is can be well in Java Swing Programming Language. Therefore Java Swing is chosen for the GUI development.

D. Naming Conventions

Naming conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier, for example, whether it's a constant, package, or class which can be helpful in understanding the code. The conventions given in this section are high level. The naming rules for the identifiers are explained below:

- **Classes**: Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep class names simple and descriptive. Use whole words and avoid acronyms and abbreviations. Examples: Class ParseInputFile
- **Methods**: Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized. Examples: ParseOIDfile()
- **Variables**: Variable names should be short yet meaningful. The choice of a variable name should be mnemonic that is, designed to indicate to the casual observer the intent of its use.
- **Constants**: The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_").Examples: int MIN_WIDTH = 4;
- **Global Variables.** Global variables should be prefixed with a 'g' and it's important to know the scope of a variable.
- **Pointer Variables**. Pointers should be prefixed by a 'p' in most cases.

E. Implementation Steps

- Run RMI batch file to start the server. we have to activate the packages of the server using RMI techniques(registry), router maintenance , packet marking source, leaf router and transition router and destination , client run all these batch files
- Run the router maintance module. In the router maintance module there are three routers namely. R101, R102 and R103. (This is the leaf, intermediate and transition router respectively). We can select any one of the router which can be connected or disconnected and select the submit option.
- In the packet marking source we give the source id as client-1 and destination id as client-4. The constructed path will be automatically created based on the data available in the data base.
- Then we select a text file which is divided into number of packets and marked with predefined value and marking probability. This is implemented in java programming using the functions
- using this logic
- X =(Math.random()*9)+31;
- int ch1=(int) (X);
- String pcnt=pktcnt+inc;
- Tpn t =(Tpn)Naming.lookup("//192.168.0.55/Server");
- t.pktEncode1(pcnt,ch1);
- String str1=Integer.toBinaryString(ch1);
- Then the packets are sent to the leaf router by selecting the send option.
- At the leaf router, the packets are received and it is checked for the values that is encoded form the packet marking source. Then it is forwarded to the intermediate process router.
- The introduction of structured programming in the 1960's and 70's brought with it the concept of Structured Flow Charts. In addition to a standard set of symbols, structured flow charts specify conventions for linking the symbols together into a complete flow chart. The structured programming paradigm evolved from the mathematically proven concept that all problems can be solved using only three types of control structures. They are as follows
- Sequence
- Decision (Selection)
- Iterative (or looping).
- Process Flow Chart

A flowchart is a common type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Data is represented in these boxes, and arrows connecting them represent flow / direction of flow of data. Flowcharts are used in analyzing, designing, documenting or managing a process or program. The flow chart for this project is as shown in the fig 1.

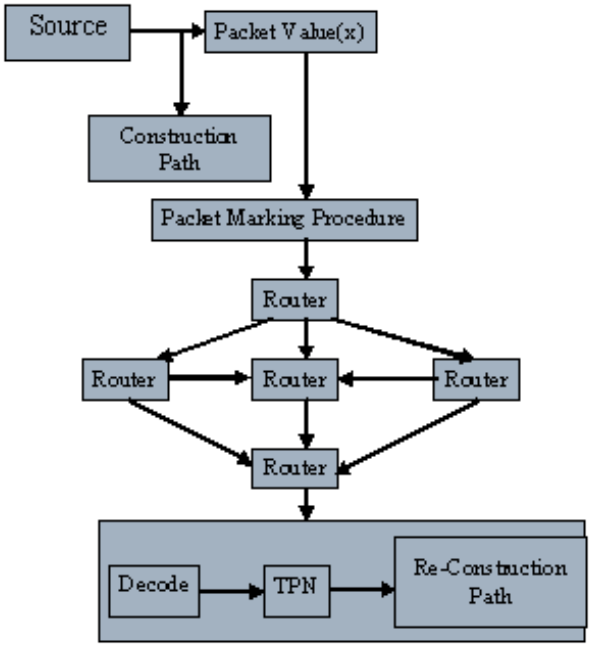

Figure 1 Flow Chart for the project

Router Maintenance

This function will create a database where the data about the routers are maintained it states whether a particular router is connected or disconnected.

- Purpose

The main purpose of this method is to check whether a router is available or not.

- Functionality

It updates the database about the availability or non-availability of router.

The various steps in this method are as below

Step 1: Start

Step 2: User can either select a particular router to be connected or disconnected.

Step3: User can submit his option to the database.

Step 4: stop.

- Input

Nothing but choosing the availability of router.

- Output

Checking the database for the options chosen by the user.

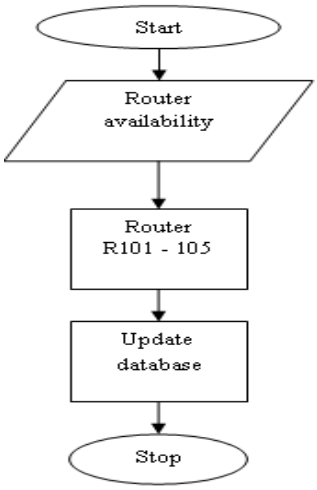

Fig 2 Flow chart for the Router Maintenance Module.

IV. SOFTWARE TESTING

Testing is the major quality control measure used during software development. Its basic function is to detect errors in the software. During requirement analysis and design the output is a document that is usually textual and not executable. After the coding phase computer programs are available that can be executed for testing purpose. This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during the previous phases. Thus the goal of testing is to uncover requirements, design and coding errors in the programs.

A. Testing Process

The basic goal of the software development process [8] is to produce that has no errors or very few errors. In an effort to detect errors soon after they are introduced, each phase ends with a verification activity such as a review. However, most of these verification activities in the early phases of software development are based on human evaluation and cannot detect all the errors. The testing process starts with a test plan. The test plan specifies all the test cases required. Then the test unit is executed with the test cases. Reports are produced and analyzed. When testing of some units is complete, these tested units can be combined with other untested modules to form new test units. Testing of any unit involves the following

- Select test cases
- Execute test cases and
- Evaluate the results of testing.

B. Test Plan

Testing is an extremely critical and time consuming activity. It requires proper planning of the overall testing process. Testing process starts with a test plan that identifies all the testing related activities that must be performed and specifies the schedule, allocates the resources and specifies guidelines for testing.
Testing can be divided into two phases:

1. Unit Testing
2. Integration Testing

The test cases are selected to ensure that the system behavior can be examined in all possible combinations of conditions. Accordingly, expected behavior of the system under different combinations is given. Therefore test cases are selected which have inputs and the outputs are on expected lines, inputs that are not valid and for which suitable messages must be given and inputs that do not occur very frequently which can be regarded as special cases.

Apart from unit testing and integration testing Regression testing must be given equal importance. The test plan should always provision regression testing in it. Regression testing is any type of software testing that seeks to uncover software errors by partially retesting a modified program. The intent of regression testing is to provide a general assurance that no additional errors were introduced in the process of fixing other problems. Regression testing is commonly used to test the system efficiently by systematically selecting the appropriate minimum suite of tests needed to adequately cover the affected change. Common methods of regression testing include rerunning previously run tests and checking whether previously fixed faults have re-emerged.

Test Environment

This section discusses in detail the environment for conducting the testing process. The Minimum hardware and software requirements are specified as follows.

Hardware requirements

- Intel Pentium IV Processor,
- 256 MB RAM,

- 20 GB HDD.
- Input and Output Devices-keyboard, Mouse and monitor

Software requirements

- Operating System: Windows XP with SP2,
- Programming Language: JDK 1.6,

The Test Setup

The steps to be followed prior to conducting the tests are as follows

- Install JDK 1.6 on the computer.
- Place the project folder in any of the hard disk drives.
- Select the main file of the project and execute it by double clicking on it.
- The project file should be in the debug mode.

After performing the above steps the test cases mentioned in the following section can be executed

C. Unit Testing

Here all the individual modules identified are tested one at a time as they are being developed. Unit testing deals with checking every single component of the code individually to make sure that the modules work properly when they are integrated.

D. Test Strategy

The strategies used to perform unit testing are described below:

- **Features to be tested** – The features to be tested, most importantly includes the accuracy of the individual unit and also the range of inputs for which the unit functions properly.
- **Items to be tested** – The items to be tested include all the individual units or functions, which collectively form the whole system In case of unit testing the items to be tested, are the individual units. Sample input, which can be any valid input for the unit and its corresponding expected output and the actual output.
- **Purpose of testing** – To examine whether the protocol anomaly detection modules are able to detect any anomaly which may be present in the Headers of the Individual Protocols
- **Pass/Fail Criteria** – The pass or fail criteria is the matching of the expected and the actual outputs of the integrated modules.

E. Integration Testing

Integration testing deals with checking whether the components work correctly in synchronization, when they are integrated. Unit testing establishes that each component is singularly working as expected. Thus integration testing is reduced to just checking the co-ordination amongst the tested components. This integration process involves building the system and testing the resultant system for problems that arise from component interactions. Integration tests should be developed from the system specification and integration testing should begin as soon as usable versions of some of the system components are available.

The main difficulty that arises in integration testing is localizing errors that are discovered during the process. These are complex interactions between the system components and, when an anomalous output is discovered, it may be hard to find the source of the error. To make it easier to locate the errors, an incremental approach to system integration and testing must be used.

F. Test Cases for Integration Testing

In this project all the methods are defined in the single class. An integration testing is done by testing the main module which invokes all the methods.

- **Integration test case for Sender and Receiver module**

V. CONCLUSION

The conclusion that can be drawn from this project is the learning of detection of the error which is the loss of packets of data in network system. Secondly, the introduction of the hackers at the intermediate router by which the path of the packet of data that has traversed can be traced and the location of the hackers is introduced by this can be found. This projected is extended by implementing with the multiple attacker environments.

REFERENCES

1. Detecting and Preventing IP-spoofed Distributed DoS Attacks, Yao Chen1, Shantanu Das1, Pulak Dhar2, Abdul motaleb El Saddik1, and Amiya Nayak, School of Information Technology and Engineering, University of Ottawa, International Journal of Network Security, Vol.7, No.1, PP.70–81, July 2008.
2. On Improving an Algebraic Marking Scheme for Detecting DDoS Attacks Moon-Chuen Lee1, Yi-Jun He and Zhaole Chen The Chinese University of Hong Kong, Dept. of Computer Science & Engineering, Journal of Information Assurance and Security -2008

3. A Precise Termination Condition of the Probabilistic Packet Marking Algorithm, Tsz-Yeung Wong, Man-Hon Wong, and Chi-Shing (John) Lui, Senior Member, IEEE, IEEE Transactions on Dependable and Secure Computing, vol. 5, no. 1, January-March 2008.
4. A proposal for new marking scheme with its performance evaluation for IP Traceback **,** S. Malliga and Dr. A. Tamilarasi , Department of Computer Science and Engineering , Kongu Engineering College , Perundurai, Erode, WSEAS Transactions on Computer Research, Issue 4, Volume 3, April 2008.
5. Trends in Denial of Service Attack Technology, CERT® Coordination Center, Kevin J. Houle, CERT/CC George M. Weaver, CERT/CC. 2008.
6. Toward a Practical Packet Marking Approach for IP Traceback Chao Gong and Kamil Sarac Department of Computer Science, University of Texas at Dallas, 2008.
7. Markov Chain Modelling of the Probabilistic Packet Marking Algorithm, Tsz-Yeung Wong, John Chi-Shing Lui, and Man-Hon Wong, Department of Computer Science and Engineering, The Chinese University of Hong Kong, International Journal of Network Security, Vol.5, No.1, PP.32–40, July 2007.
8. Rajib Mall, "Fundamentals of Software Engineering", Prentice-hall Of India Pvt Ltd Publishing, 3rd edition, 2009.