



A NOVEL IP TRACE BACK FOR DDOS ATTACKS USING ZOMBIES BASED ON ENTROPY VARIATION

J.VIJAYALAKSHMI, M.C.A., M.Phil.,
Department of Computer Science
Bharathidasan University Model College
Vedaraniyam

Abstract— A zombie is a computer, connected to the internet that has been compromised by a cracker. Zombie computers are often used to launch distributed denial-of-service attacks. Distributed denial-of-service (DDoS) attacks are a critical threat to the internet. However, the memory less feature of the internet routing mechanisms makes it extremely hard to trace back to the source of these attacks. As a result, there is no effective and efficient method to deal with this issue so far. In this paper, I propose a novel trace back method for ddos attacks that is based on entropy variations between normal and ddos attack traffic, then compare this variation with the router's buffer's capacity, which is fundamentally different from commonly used packet marking techniques.

Keywords—IP Trace back, DDOS, Zombie, Entropy Variation

I. INTRODUCTION

IT is an extraordinary challenge to traceback the source of Distributed Denial-of-Service (DDoS) attacks in the Internet. In DDoS attacks, attackers generate a huge amount of requests to victims through compromised computers (zombies), with the aim of denying normal service or degrading of the quality of services. The key reason behind this phenomenon is that the network security community does not have effective and efficient traceback methods to locate attackers as it is easy for attackers

to disguise themselves by taking advantages of the vulnerabilities of the World Wide Web, such as the dynamic, stateless, and anonymous nature of the Internet. IP trace back means the capability of identifying the actual source of any packet sent across the Internet. Because of the vulnerability of the original design of the Internet, we may not be able to find the actual hackers at present. In fact, IP traceback schemes are considered successful if they can identify the zombies from which the DDoS attack packets entered the Internet. Research on DDoS detection mitigation and filtering has been conducted pervasively. However, the efforts on IP traceback are limited.

A number of IP traceback approaches have been suggested to identify attackers and there are two major methods for IP traceback, the probabilistic packet marking (PPM) and the deterministic packet marking (DPM). Both of these strategies require routers to inject marks into individual packets. Moreover, the PPM strategy can only operate in a local range of the Internet (ISP network), where defender has the authority to manage. However, this kind of ISP networks is generally quite small, and we cannot traceback to the attack sources located out of the ISP network.

The DPM strategy requires all the Internet routers to be updated for packet marking. However, with only 25 spare bits available in as IP packet, the scalability of DPM is a huge problem. Moreover, the DPM mechanism poses an

extraordinary challenge on storage for packet logging for routers. Therefore, it is infeasible in practice at present. Further, both PPM and DPM are vulnerable to hacking, which is referred to as packet pollution.

II. RELATED WORK

A. Existing System

It is obvious that hunting down the attackers (zombies), and further to the hackers, is essential in solving the DDoS attack challenge. In general, the traceback strategies are based on packet marking. Packet marking methods include the PPM and the DPM. The PPM mechanism tries to mark packets with the router's IP address information by probability on the local router, and the victim can reconstruct the paths that the attack packets went through.

The PPM method is vulnerable to attackers, as attackers can send spoofed marking information to the victim to mislead the victim. The accuracy of PPM is another problem because the marked messages by the routers who are closer to the leaves (which means far away from the victim) could be overwritten by the downstream routers on the attack tree. At the same time, most of the PPM algorithms suffer from the storage space problem to store large amount of marked packets for reconstructing the attack tree.

The deterministic packet marking mechanism tries to mark the spare space of a packet with the packet's initial router's information, e.g., IP address. Therefore, the receiver can identify the source location of the packets once it has sufficient information of the marks. The major problem of DPM is that it involves modifications of the current routing software, and it may require very large amount of marks for packet reconstruction. Moreover, similar to PPM, the DPM mechanism cannot avoid pollution from attackers. Savage et al. first introduced the probability-based packet marking method, node appending, which appends each node's address to the end of the packet as it travels from the attack source to the victim.

B. Proposed System

IP traceback methods should be independent of packet pollution and various attack patterns. In our previous work on DDoS attack detection, we compared the packet number distributions of packet

flows, which are out of the control of attackers once the attack is launched, and we found that the similarity of attack flows is much higher than the similarity among legitimate flows, e.g., flash crowds. Entropy rate, the entropy growth rate as the length of a stochastic sequence increases, was employed to find the similarity between two flows on the entropy growth pattern, and relative entropy, an abstract distance between two probabilistic mass distributions, was taken to measure the instant difference between two flows.

The proposed strategy is fundamentally different from the existing PPM or DPM traceback mechanisms, and it outperforms the available PPM and DPM methods. Because of this essential change, the proposed strategy overcomes the inherited drawbacks of packet marking methods, such as limited scalability, huge demands on storage space, and vulnerability to packet pollutions.

The implementation of the proposed method brings no modifications on current routing software. Both PPM and DPM require update on the existing routing software, which is extremely hard to achieve on the Internet. On the other hand, this method can work independently as an additional module on routers for monitoring and recording flow information, and communicating with its upstream and downstream routers when the pushback procedure is carried out.

This method will be effective for future packet flooding DDoS attacks because it is independent of traffic patterns. Some previous works depend heavily on traffic patterns to conduct their traceback. For example, they expected that traffic patterns obey Poisson distribution or Normal distribution. However, traffic patterns have no impact on the proposed scheme; therefore, we can deal with any complicated attack patterns, even legitimate traffic pattern mimicking attacks.

This method can achieve real-time traceback to attackers. Once the short-term flow information is in place at routers, and the victim notices that it is under attack, it will start the traceback procedure. The workload of traceback is distributed, and the overall traceback time mainly depends on the network delays between the victim and the attackers.

III. PROJECT DEFINITION

In this Proposed System of implementation, the Flow Entropy is measured to find the Attack. We

categorize the packets that are passing through a router into flows. A flow is defined by a pair—the upstream router where the packet came from and the destination address of the packet. Entropy is an information theoretic concept, which is a measure of randomness we employ entropy variation in this paper to measure changes of randomness of flows at a router for a given time interval. In this project all the incoming flows as input flows and all the flows leaving from the router are named as output flows. In this Process, the buffer level of every Router is calculated and compared with the input Flows and with the Output Flows. The paper means to find only the Flow Entropy as the main Point of Identifying the Attacker. But this includes, the Comparison of Flow Entropy with the Buffer Level of every Router.

IV. PROBLEM STATEMENT

This method can achieve real-time traceback to attackers. Once the short-term flow information is in place at routers, and the victim notices that it is under attack, it will start the traceback procedure. The workload of traceback is distributed, and the overall traceback time mainly depends on the network delays between the victim and the attackers.

A. Network Construction

This module is developed in order to create a dynamic network. In a network, nodes are interconnected with the admin, which is monitoring all the other nodes. All nodes are sharing their information with each others.

B. Attackers

The attacker first establishes a network of computers that will be used to generate the huge volume of traffic needed to deny services to legitimate users of the victim. To create this attack network, attackers discover vulnerable hosts on the network. DDoS attack, the master computer orders the zombies to run the attack tools to send huge volume of packets to the victim.

C. Flow Entropy Variation

We categorize the packets that are passing through a router into flows. A flow is defined by a pair the upstream router where the packet came from, and the destination address of the packet. Entropy is an information theoretic concept, which is a measure of randomness. We employ entropy variation in this paper to measure changes of

randomness of flows at a router for a given time interval. We notice that entropy variation is only one of the possible metrics. Change point of flows, to identify the abnormality of DDoS attacks however, attackers could cheat this feature by increasing attack strength slowly. We can also employ other statistic metrics to measure the randomness, such as standard variation or high-order moments of flows.

D. IP Trace back

Another defensive countermeasure is trace back, which enables law enforcement agencies to identify the original worm propagators and punish them. A trace back scheme typically involves a number of routers, which monitor all through-traffic and store traffic logs in a storage server. When a “trace back” order is given, the traffic logs (e.g., flow-level recorded logged by the networks) are postmortem analyzed in order to identify the origins of the worm propagator. When the source of the worm is detected the system alerts the node about the source and blocks all packets from that particular source.

E. Removal of IP Source

Once we apply the IP Trace back system, we can identify the exact source of the system which is involved in spreading of the worms. We are identifying the Source of the Worm creator & we can eliminate that system from the network. This process of elimination would create more secured communication.

V. TESTING

Testing is a schedule process carried out by the software development team to capture all the possible errors, missing operations and also a complete verification to verify objective are met and user requirement are satisfied. The design of tests for software and other engineering products can be as challenging as the initial design to the product itself.

A. Testing Expertise of Library

Each increment has been positioned under each phase of the “l model” and each individual increment is analyzed, designed, coded, and tested. Then all the core products are integrated and once again the whole product is passed out under the phases any the wholesome core product is delivered to the inmates

as the e-mail. As after deliveries service the system users are analyzed for any other user expectation. In order to back up the testing process of the e-mail, the assessment is used. The typical objectives of the e-mail are to formally document the project plan for: • test philosophy, test works product, test teams, test task and test tools.

B. System Testing

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operations commences.

- Testing is the process of correcting a program with intend of fining an error.
- A good test case is one that has high probability of finding a yet undiscovered error.
- A successful test is one that uncovers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are subjected to variety of test on-line response, volume, stress, recovery and security and usability tests. A series of test are performed before the system is ready for user acceptance testing.

C. Unit Testing

In this testing we test each module individual and integrated the overall system. Unit testing focuses verification efforts on the smaller units of software design in the module. This is also known as 'Module' testing. The modules of the system are tested separately. The testing is carried out during programming stage itself. In this testing step each module is found to working satisfactory as regard to the expected output from the module. There are some validation checks for verifying the data input given by the user which both the formal and validity of the entered. It is very easy to find error debug the system.

- It enables various product delivery teams to understand what is expected of them without the down fall of error.
- Enable the identification of unit level defects by causing corresponding failures

to during each run of the e-mail application.

- Identify defects that are not easily identified during other kinds of testing.
- Enable the developer to confidently iterate and refactor the software unit of the e-mail to database connection repository, knowing that any defect will be quickly discovered during regression testing.
 - Enable the library developer to determine if the unit
 - Is complete
 - Fulfills its responsibilities and does not violate its assertions.
 - Work as designed
 - Is ready to be submitted for integration as part of a code build.

D. Integration Testing

Data can be lost across an interface; one module can have an adverse effort on the other sub functions when combined may not produced the desired major functions. Integrated testing is the systematic testing for constructing the uncover error within interface. This testing was done with sample data. The developed system has run successful for this data. The need for integrated test is to find the overall system performance.

E. Validation Testing

The software is completely assembled as a package, interface error has been uncovered and corrected and final series of software test, validation test begins. Validation testing can be defined many was but a simple definition is that validation. Succeeds when the software function in a manner that can be reasonably accepted by the customer .After validation test has been conducted one of the two possible condition exists.

- The function or performance characteristics confirm to specification and are accepted.
- A deviation from specification is uncovered and a deficiency list is created enables the developers to understand how

F. Acceptance Testing

User Acceptance of the system is the key factor for the success of the system. The system consideration is tested for user acceptance by constantly keeping in touch with prospective system

at the time of developing and making change whatever required.

This is done with regard to the following points.

- Output screen design
- Input screen design
- Menu driven system

G. Testing Overview

Testing presents an interesting challenge for the software engineer. During the earlier definition and development phases, the engineer attempts to build software from an abstract concept to an acceptable implementation. In testing, the engineer creates a series of test cases that are intended to demolish the software that has been built. Testing requires that the developer discard preconceived notions of the correctness of the software developed and overcome a conflict of interest that occur when error are uncovered.

H. Testing Objectives

There are several rules that can serve as testing objectives. They are as follows:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

VI. IMPLEMENTATION

System implementation is a stage in the project where the theoretical design is turned into working system. It is the process of converting a new system into operation. The implementation phase of software development is concerned with translating design specification into source code. The most crucial stages is giving the users confidence that the new system will work effectively and efficiently.

- Step 1: construct the network, set one node as admin.
- Step 2: send a file from source to destination,
- Step 3: the file will be converted in to packets; these packets will be transmitted across the network.

- Step 4: If the packet is dropped, then IP trace back is initiated automatically.
- Step 5: if the attacker is found, that node should be removed
- Step 6: this information is intimated to the sender
- Step 7: finally, there is a secure communication between sender and receiver.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

VII. SOFTWARE REQUIREMENT SPECIFICATIONS

Requirement Analysis is a software engineering task that bridges the gap between system level software allocation and software design. It provides the system engineer to specify software function and performance indicate software's interface with the other system elements and establish constraints that software must meet.

The basic aim of system requirements is to obtain a clear picture of the needs and requirements of the end-user and also the organization. Analysis involves interaction between the clients and the analysts usually analysts research a problem from any questions asked and reading existing documents. The analysts have to uncover the real needs of the user even if they don't know them clearly. During analysis it is essential that a complete and consistent set of specifications emerge for the system. Here it is essential to resolve the contradictions that could emerge from information got from various parties.

This project will be implemented using JAVA and MYSQL Server. This chapter provides an insight into each of these that are used in our system and concludes with a few snapshots of the final GUI.

VIII. SYSTEM DESIGNS

A. System Architecture

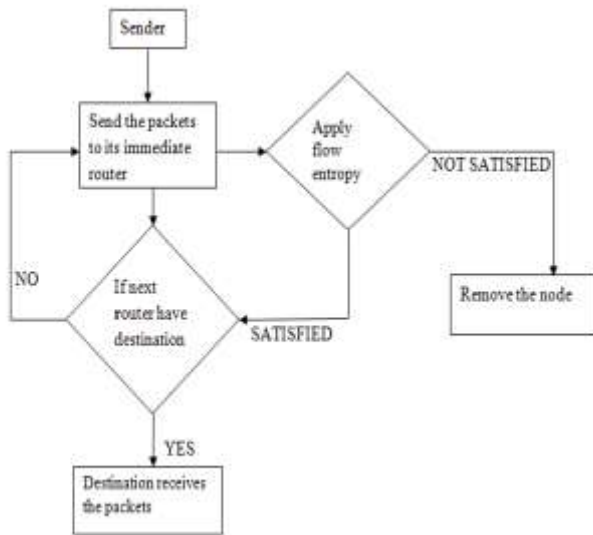


Fig. 1. System Architecture

IX. DATA FLOW DIAGRAM

A. Without Attackers

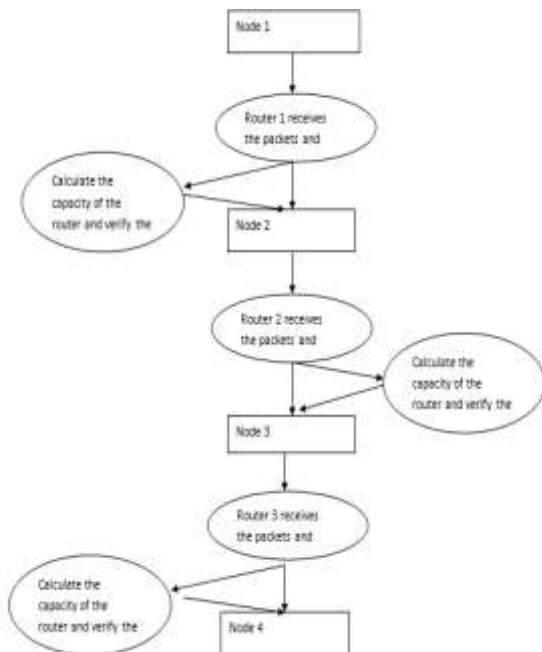


Fig. 2. DFD without Attackers

B. With Attackers

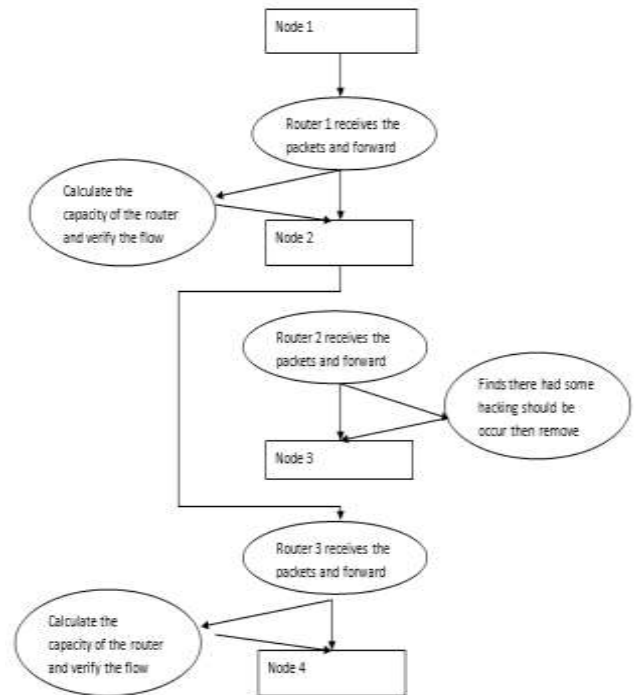


Fig. 3. DFD with Attackers

X. MAINTENANCE

Maintenance covers a wide range of activities, including correcting coding, design errors, updating documentation and test data, and upgrading user support. Many activities classified as maintenance or actually enhancement. Maintenance means restoring something to its original conditions. Unlike hardware, however software does not wear out, it is corrected. In contract, enhancement means adding, modifying or redeveloping the code to support changes in the specification.

XI. CONCLUSION

This project focuses on effective and efficient trace back scheme against DDos attacks based on entropy variations. It is fundamentally different from currently adopted packet marking strategies. On the other hand, it needs no marking on packets, and therefore, avoids the inherent shortcomings of packet marking mechanisms.

It employs the features that are out of the control of hackers to conduct the IP trace back. Moreover, this model can work as an independent software module with current routing software.

This makes it a feasible and easy to be implemented solution for the current Internet.

XII. FUTURE WORK

This paper did not consider the flash crowds. This method may treat flash crowd as a DDOS attack and therefore resulting in false positive alarms. This can be further enhanced by considering this issue.

REFERENCES

- [1] Shui Yu, Wanli Zhou, Weijia jia, Member, IEEE Computer Society, "Traceback of DDOS Attacks Using Entropy Variations.", vol. 22, no. 3, march 2011.
- [2] C. Patrikakis, M. Masikos, and O. Zouraraki, "Distributed Denial of Service Attacks," *The Internet Protocol J.*, vol. 7, no. 4, pp. 13-35, 2004.
- [3] Y. Kim et al., "PacketScore: A Statistics-Based Packet Filtering Scheme against Distributed Denial-of-Service Attacks," *IEEE Trans. Dependable and Secure Computing*, vol. 3, no. 2, pp. 141-155, Apr.-June 2006.
- [4] Y. Chen and K. Hwang, "Collaborative Detection and Filtering of Shrew DDOS Attacks Using Spectral Analysis," *J. Parallel and Distributed Computing*, vol. 66, pp. 1137-1151, 2006.
- [5] M.T. Goodrich, "Probabilistic Packet Marking for Large-Scale IP Traceback," *IEEE/ACM Trans. Networking*, vol. 16, no. 1, pp. 15-24, Feb. 2008.
- [6] T.K.T. Law, J.C.S. Lui, and D.K.Y. Yau, "You Can Run, But You Can't Hide: An Effective Statistical Methodology to Traceback DDOS Attackers," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 9, pp. 799-813, Sept. 2005.
- [7] S. Savage, "Network Support for IP Traceback," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 226-237, June 2001.
- [8] Belenky and N. Ansari, "IP Traceback with Deterministic Packet Marking," *IEEE Comm. Letters*, vol. 7, no. 4, pp. 162-164, Apr. 2003.