Volume 3, Issue 10, October 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Countering Varying DoS Attacks using Snort Rules

Mothalla Saritha[1], [1] Student, DEPT.OF CSE, V.R. Siddhartha Engineering College (Autonomous), Kanuru, Vijayawada, India

Mukesh Chinta[2], [2] Asst.Professor, DEPT.OF CSE, V.R. Siddhartha Engineering College (Autonomous), Kanuru, Vijayawada, India

Abstract: Distributed denial of service (DDoS) attack works to shut down a particular victim web server with packet drencher. DDoS attacks evolved from relatively humble megabit inception in 2000 to the largest recent DDoS attacks breaking the 100 GB/s barrier which reveals that the majority of ISPs (Internet Service Provider) today lack with an appropriate infrastructure to lessen them. The sudden increase in internet traffic can stimulate the server to reduce the performance. My Doom devastation on micro soft and wiki leaks are some of the examples who had already confronted and experienced DDoS attacks whereas other major Internet players like Amazon, CNN and Yahoo are also not in exception. Even though it is challenging task, ethical hacker should reveal the early discovery of these attacks to protect victim server's network infrastructure resources which is crucial in state-of-the intrusion detection systems. Exisitng intrusion prevention systems like FireCol is efficient in thwarting DDoS but its architecture is based on ISP collaboration and virtual protection rings. In this paper, authors propose to use an IPS rule (Snort rules) based DDoS detection approach which checks both data and header packets. This enables the detection system to eliminate other forms of DoS attacks like Slow Read DoS attack. Its effectiveness with low overhead and support for incremental deployment in real networks are demonstrated in experimental results.

Index Terms: Snort Rule Structures, FireCol, IDS, IPS, packet, network communication, DDOS.

I. INTRODUCTION

DDoS attacks are mainly used for flooding a particular victim with massive traffic and paralyzing its services [4]. Recent works aim at countering DDoS attacks by fighting the underlying vector, which is usually the use of botnet. A botnet is a large network of compromised machines (bots) controlled by one entity (the master). The master can launch synchronized attacks such as DDoS by sending orders to the bots via a Command & Control channel [2][3] . Unfortunately, detecting a botnet is hard and efficient solution which require scanning entities to participate actively in the botnet itself unlike entities scanning from a safe distance [6]. A single intrusion prevention system (IPS) or intrusion detection system (IDS) can hardly detect such DDoS attacks, unless they are located very close to the victim. The IDS/IPS may crash because it needs to deal with an overwhelming volume of packets (some flooding attacks reach 10–100 GB/s) even in latter case. Moreover, allowing huge traffic to transit through the Internet and only detect/block it at the host IDS/IPS may severely strain the Internet resources[5] [7]. Hence, a collaborated system which can empower the single host based detection and blocking procedures are required for efficient prevention of DDoS. To overcome such problems, a new collaborative system called FireCol was proposed to detect flooding DDoS attacks as far as possible from the victim host and as close as possible to the attack source(s) at the Internet service provider (ISP) level. [3][6] FireCol relies on a distributed architecture composed of multiple ISPs forming overlay networks of protection rings around subscribed customers. The virtual rings use horizontal communication when the degree of a potential attack is high[2]. In this way, threat is measured based on the overall traffic bandwidth directed to the customer compared to the maximum bandwidth it supports.

FireCol consists of

- Packet Processor
- Metrics Manager
- Selection Manager
- Score Manager
- Collaboration Manager

FireCol architecture uses the following algorithms: Packet rate computation using rule frequencies (collaboration manager) and Mitigation Shields Deployment. In addition to detect flooding DDoS attacks, FireCol also helps in detecting other flooding scenarios such as flash crowds and other botnet-based DDoS attacks which offers an enhanced performance. But, FireCol's defense procedures require different ISP's collaboration to form virtual protection rings which has real time implementation issues involving total revamp of the architecture[14]. FireCol's defense procedures (virtual protection rings notion) are not based on IPS rule structures (Snort Rules). In this paper, the proposed system is extending FireCol to support different IPS rule structures which will help FireCol thwart and other forms of DoS attacks like entrant Slow Read DoS attack. Proposed system was Snort's detection system which is based on rules.

Like viruses, most intruder activity has some sort of signature. Information about these signatures is used to create Snort rules. These rules in turn are based on intruder signatures. Snort rules can be used to check various parts of a data packet apart from header scanning adapted by prior approaches. A rule may be used to generate an alert message or in terms of Snort pass the data packet, i.e., drop it silently. Thus enabling a detection system eliminating other forms of DoS attacks such as Slow Read DoS attack. Snort Based DoS detection system can be a real time efficient and feasible implementation that can counter varying DoS attack forms.

II. RELATED WORK

In [10] a flow connection entropy (FCE) series is used to reduce the dimensionality of the high-dimensional traffic sequence, and then use the time-series decomposition method to divide the FCE series into a trend component and a steady random component. It then applies a double autocorrelation technique and an improved cumulative sum technique to the trend and random components, respectively, to detect anomalies in both long-term and short-term trends in the traffic. Pros are by calculating the distribution of the FCE series, a coarse-grained estimation of the traffic is obtained. Compare to traditional attacks, DDoS attacks are more dangerous because it is harder to detect, especially when embedded in a large amount of traffic. In [11] The statistics of normal SYN arrival rate (SAR) and confirm it follows normal distribution. It identifies the attack by testing 1) The difference between incoming SAR and normal SAR, and 2) The difference between the number of SYN and ACK packets. This SYN flooding attacks are divided into three main approaches: packet marking, proactive and reactive. Packet marking approach marks the suspicious packets with some bits at distributed routers, and then filters them in case of violating rule or exceeding threshold. Some IP Traceback schemes employ packet marking to trace attacks that use source address spoofing.

Proactive approach usually 1) modifies the existing protocol to prevent DDoS attack from happening or 2) differentiates malicious traffic from normal one with the use of Packet Score or test probability or with the deployment of intelligent framework. On the other hand, reactive approach takes some response after the malicious traffic is detected. Much effort has been concentrated on reactive approach. PROS are these method detects low-rate attack by simply testing the difference between the number of SYN and ACK packets. It can effectively differentiate between normal and flooding traffic. In [12], A novel scheme for detecting and preventing the most harmful and difficult to detect DDoS Attack, that use IP address spoofing to disguise the attack flow. This scheme is based on a firewall that can distinguish the attack packets from the packets sent by users, and thus filters out most of the attack packets before they reach the victim. It allows the firewall system to config- ure itself based on the normal traffic of a Web server, so that the occurrence of an attack can be quickly and precisely detected. Pros are firewall can effectively and efficiently differentiate between good and bad packets under attack. The deployment cost of our scheme is very low.

In [13] a distance-based distributed DDoS defense framework is one that defends against attacks by coordinating between the distance-based DDoS defense systems of the source ends and the victim end. Distance-based defense system has three major components: detection, traceback, and traffic control. In the detection component, two distance-based detection techniques are employed. For the traceback component, the existing Fast Internet Traceback (FIT) technique is employed to find remote edge routers which forward attack traffic to the victim.In distance -based rate limit mechanism, traffic control component at the victim end requests the source-end defense systems to set up rate limits on these routers in order to efficiently reduce the amount of attack traffic. Pros are the average distance estimation DDoS detection technique works on distance values directly. Using distance and history information to calculate the rate limit is a unique contribution. cons are Legitimate traffic cannot utilize resources quickly and effectively during recovery.The distance-based DDoS defense framework does not perform well to decide whether an attack has finished or not.

In [14] *FireCol,* a new collaborative system that detects flooding DDoS attacks as far as possible from the victim host and as close as possible to the attack source(s) at the Internet service provider (ISP) level. FireCol relies on a distributed architecture composed of multiple IPSs forming overlay networks of protection rings around the subscribed customers. Pros are in a collaborative approach besides the victim server its intermediate servers in protection rings take part in thwarting DDoS attacks, thus reducing load on victim server. Experimental analysis shows good performance and robustness of *FireCol* and highlighted good practices for its configuration. But, FireCol was designed in single IPS Rule structure. This project proposes to use an different intrusion prevention system rule structures.

III. BACKGROUND

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level[2][3]. Usually, an intrusion detection system captures data from the network and applies its rules to data or detects anomalies in it. Input plug-ins are present to detect anomalies in protocol headers because the snort is a "rule based" IDS. Snort uses rules stored in text files that can be modified by a text editor. Rules are grouped in categories. [6][8] Rules belonging to each category are stored in separate files. Snort reads these rules at the start-up time and builds internal data structures or chains to apply these rules to captured data. [4] Finding signatures and using them in rules is a tricky job because the more rules use, the more processing power is required to process captured data in real time. [2] It is important to implement as many signatures as it can use as few rules as possible. Snort comes with a rich set of pre-defined rules to detect intrusion activity and it is free to add own rules at will. To avoid false alarms, built-in rules can also remove.

IV. PROPOSED SYSTEM

SNORT is one of the most popular Network intrusion detection system(NIDS).SNORT is Open Source, allows people to contribute and analyze the program construction. SNORT uses the most common open-source license known as

the GNU (General Public License). Snort is logically divided into multiple components. These components work together to detect particular attacks and to generate output in a required format from the detection system. Snort's architecture consists of four basic components:

- The sniffer
- The preprocessor
- The detection engine
- The output

Packet Sniffer

A packet sniffer is a device used to tap into networks. It works in a similar fashion to a telephone wiretap. Nevertheless, it is used for data networks instead of voice networks. A network sniffer allows an application or a hardware device to eavesdrop on data network traffic. But in local LANs and legacy networks it can be other protocol suites such as IPX and AppleTalk traffic, where as in the case of the Internet it consist of the IP Traffic. Packet sniffers have various uses:

- Network analysis and troubleshooting.
- Performance analysis and benchmarking
- Eaves dropping for clear-text passwords and other interesting tidbits of data.

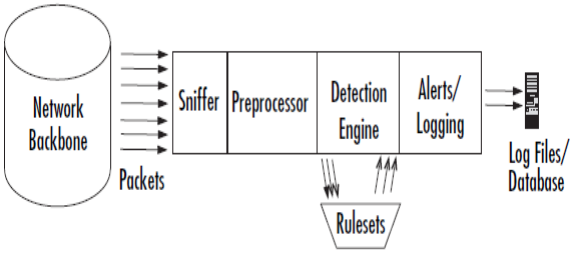

Fig 1: Snort Architecture

Preprocessor

A preprocessor take the raw packets and checks them against certain plug-ins. The behavior of the packet is checked by the certain type of plug-in like RPC plug-in, a port scanner plug-in and an HTTP plug-in. Once the packet is determined to have a particular type of "behavior," it is then sent to the detection engine. Snort supports many kinds of preprocessors and their attendant plug-ins. These plug-ins covers many commonly used protocols as well as larger-view protocol issues such as IP fragmentation handling, flow control, port scanning and deep inspection of richly featured protocols.

Step 1: Extend the Original Rule set{R}

Step 2: Initially Extended Rule Set E=ϕ, then E= Insert (R_i, E).

Step 3: For all Rule structure E_r from E

Step 4: Calculating the each matching rules in Original Rule Set Matching Rules i.e. Mi = MxU Mi; where Mi is super rule set and Mx is sub rule set.

Step 5: We repeat the above 2,3,4 steps for each client present in network.

Algorithm1: Detection of rule structure algorithm.

As shown in the above algorithm1, detection with matching rule structure work as follows. Initially, we are taking original rule set R={R1,R2,….Ri} as input. Each rule set associated with match list with index provided by our original rule set. Subsequently, extended rule set scans each rule E_i in E and check the matching relations between original rule set structures with generated rule set. If matching is done in this relation then we are assigning that particular client as attacker and alert is generated.If any rule structures are not matching with original rule set then we are addin g that client into network.

Detection Engine

Once packets have been handled by all enabled preprocessors then they are handed off to the detection engine which is the meat of the signature-based IDS in Snort. The preprocessor sends the data to the detection engine, its plug-ins and that data is checked through a set of rules. If the rules match the data in the packet then they are sent to the alert processor. The signature-based IDS function is accomplished by using various rule sets. The rule sets are grouped by category (buffer overflows, Trojan horses, access to various applications) and are updated regularly.

The rules themselves consist of two parts:

■ **The rule header** The rule header is basically the action to take (log or alert), type of network packet (TCP, UDP, ICMP, and so forth), source, destination IP addresses, and ports

■ **The rule option** The option is the content in the packet that should make the packet match the rule.

The detection engine and its rules are the largest portion (and steepest learning curve) of new information to learn and understand with Snort, where Snort has a particular syntax that it uses with its rules. Rule syntax can involve the content, the type of protocol, the header, the length, and other various elements including garbage characters for defining butter overflow rules. If we want to generate new rules from existing rules it is known as generalizing SNORT rules.

Alerting/Logging Component

After the Snort data goes through the detection engine it needs to go out somewhere. If the data matches a rule in the detection engine, an alert is triggered. Depending upon "what the detection engine finds inside a packet", then the alert may be generated or activity can be log when the packet is detected. Logs are kept in simple text files, tcp dump-style files or some other form. Alerts can be sent to a log file, through a network connection, through UNIX sockets or Windows Popup (SMB), or SNMP traps. The alerts can also be stored in an SQL database such as MySQL and Postgress.

V. PERFORMANCE

Consider an Internet packet that contains a variation of a known attack, there should be some automated way to identify the packet as nearly matching a NIDS attack signature. If a particular statement has a set of conditions against it, an item may match some of the conditions. Whereas Boolean logic would give the value false to the query 'does this item match the conditions'? Our logic could allow the item to match to a lesser extent rather than not at all. This principle can be applied when comparing an Internet packet against a set of conditions in a SNORT rule. Our hypothesis is that if all but one of the conditions are met, an alert with a lower priority can be issued against the Internet packet, as the packet may contain a variation of a known attack. While implementation, generalization in the case of matching network packets against rules, involves allowing a packet to generate an alert if:

• The conditions in the rule do not all match, yet most of them do;

• The only conditions that do not match exactly nearly match.

When implementing generalized rules, the execution time was 1 second to process and convert the original 1,325 rules into a total of 6,975 rules. The generalized Content execution time was 2 seconds to process and convert the same 1,325 original rules, into a total of 18,265 rules. These execution times would easily be acceptable for most potential uses such as each time the SNORT rules were downloaded for signature updates. The increase in the number of rules affected the time spent processing network traffic data as follows:

• Using the original rules, Snort took approx 100 seconds to process 1,635,267 packets;

• Using the generalized (inverted) rules, Snort took approx 400 seconds to process the same packets.

• Using the generalized content rules, Snort took approx 1,000 seconds to process the packets. The change in SNORT's processing time is an increase of around four to ten times and roughly in line with the increase in the number of rules.

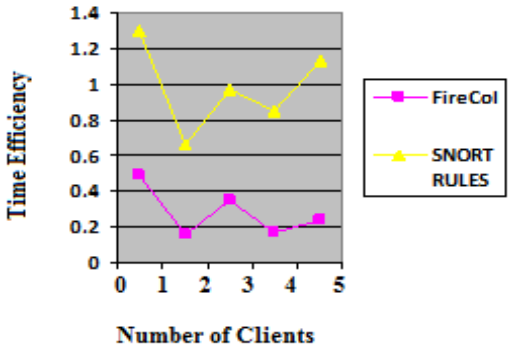

Figure 2: Time comparison results for FireCol and Snort Rule detection systems.

As shown in the above figure, it distinguishes the comparison results between both existing and proposed approaches developed in our project. In the existing approach, FireCol technique was used for detection of Denial-of-Service attacks in network communication. In this technique, no rule structure process was provided for detection of those attacks present in the network communication. In the proposed technique Intrusion detection system rules structure was developed for developing network performance with equal priority values of each node present in the network.

This section describes the network performance results using different rule structure for detection of Denial-of-Service attacks in network communication process. Different Snort rule structures like DOS, DDOS, Web-Attack, and SCAN are developed. In proposed approach DOS based rules are used and defined a java application, deployed at each node to classify incoming packets with these rules and drop the packets that are satisfied to be DOS packet.Those results were taking more time when compared to FireCol detection system. Because FireCol doesn't provide classification structure for each client in network.

VI. CONCLUSION

In this paper, the proposed system is extending FireCol to support different IPS rule structures which will help FireCol thwart and eliminate other forms of DoS attacks like Slow Read DoS attack .Snort Based DoS detection system can be a real time efficient and feasible implementation that can counter varying DoS attack forms.

REFERENCES

[1] S Axelsson (2000) 'Intrusion Detection Systems: A Survey and Taxonomy', Chalmers University Tech Report, 99-15.

[2] Proctor, Paul E. The Practical Intrusion Detection Handbook. New Jersey: Prentice Hall PTR, 2001.

[3] Northcutt, Steven. Network Intrusion Detection, An Analyst's Handbook. Indianapolis: New Riders, 1999.

[4] Bace, Rebecca. "An Introduction to Intrusion Detection and Assessment: for System and Network Security Management." ICSA White Paper, 1998.

[5] G. Badishi, A. Herzberg, and I. Keidar, "Keeping denial-of-service attackers in the dark," *IEEE Trans. Depend. Secure Comput.*, vol. 4, no. 3, pp. 191–204, Jul.–Sep. 2007.

[6] T. Peng, C. Leckie, and K. Ramamohanarao, "Detecting distributed denial of service attacks by sharing distributed beliefs," in *Proc. 8th ACISP*, Wollongong, Australia, Jul. 2003, pp. 214–225.

[7] M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, and B. Tierney, "The NIDS cluster: Scalable, stateful network intrusion detection on commodity hardware," in *Proc. 10th RAID*, Sep. 2007, pp. 107–126.

[8] Sourcefire Inc, M Roesch and C Green (2006) 'SNORT Users Manual - SNORT Release: 2.6.0', http://www.snort.org

[9] J Hoagland and S Staniford (2003) 'Viewing IDS alerts: Lessons from SnortSnarf',http://www.silicondefense.com/research/whitepapers/index.php.

[10] Haiqin Liu and Min Sik Kim School of Electrical Engineering and Computer Science Washington State University "Real-Time Detection of Stealthy DDoS Attacks Using Time-Series Decomposition",2010.

[11] Chin-Ling Chen (Department of Information Management National Pingtung Institute of Commerce), " A New Detection Method for Distributed Denial-of-Service Attack Traffic based on Statistical Test", *Journal of Universal Computer Science, vol. 15, no. 2 (2009).*

[12] Yao Chen1, Shantanu Das1, Pulak Dhar2, Abdulmotaleb El Saddik1, and Amiya Nayak1 School of Information Technology and Engineering, University of Ottawa1, "Detecting and Preventing IP-spoofed Distributed DoS Attacks" International Journal of Network Security, Vol.7, No.1, PP.70–81, July 2008.

[13] J. Françcois, A. El Atawy, E. Al Shaer, and R. Boutaba, "A collaborative approach for proactive detection of distributed denial of service attacks," in Proc. IEEE MonAM, Toulouse, France, 2007, vol. 11.

[14] Jeerome Francois, Issam Aib, Member, IEEE, and Raouf Boutaba, Fellow, IEEE," FIRECOL:A Collaborative Protection Network for the Detection of Flooding DDoS Attacks", IEEE 2012 Transaction on Networking, Volume :PP, Issue:99.

[15] D. Das, U. Sharma, and D. K. Bhattacharyya, "Detection of HTTP flooding attacks in multiple scenarios," in *Proc. ACM Int. Conf. Commun., Comput. Security*, 2011, pp. 517–522.

[16] H. Liu, "A new form of DOS attack in a cloud and its avoidance mechanism," in *Proc. ACM Workshop Cloud Comput. Security*, 2010, pp. 65–76.

[17] A. Sardana, R. Joshi, and T. hoon Kim, "Deciding optimal entropic thresholds to calibrate the detection mechanism for variable rate DDoS attacks in ISP domain," in *Proc. ISA*, Apr. 2008, pp. 270–275.

[18] I. B.Mopari, S. G. Pukale, and M. L. Dhore, "Detection of DDoS attack and defense against IP spoofing," in *Proc. ACM ICAC3*, 2009, pp.489–493.

[19] Jéerôme François, Issam Aib, Raouf Boutaba*," FireCol*: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks*",* IEEE 2012,Transaction on Networking,Volume :PP, Issue:99.

[20] [Siris and Papagalou 2004] Siris, V. A., Papagalou, F.: "Application of anomaly detection algorithms for detecting SYN flooding attacks"; GLOBECOM 2004-IEEE Global Telecommunications Conference, 1 (Dec 2004), 5013-5013.

[21] [Xiang and Zhou 2005] Xiang, Y., Zhou, W.: "Mark-aided distributed filtering by using neural network for DDoS defense"; GLOBECOM 2005-IEEE Global Telecommunications Conference, 1 (November 2005), 316-320.

[22] J. W. Haines, R. P. Lippmann, D. J. Fried, M. A. Zissman, E. Tran, and S. B. Boswell, "1999 DARPA intrusion detection evaluation: Design and procedures," Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts, U.S.A., Tech. Rep. 1062, Feb. 2001.

[23] J. Françcois, A. El Atawy, E. Al Shaer, and R. Boutaba, "A collaborative approach for proactive detection of distributed denial of service attacks," in Proc. IEEE MonAM, Toulouse, France, 2007, vol. 11.

[24] Chang and R.K.C., "Defending against flooding-based distributed denialof- service attacks: a tutorial," IEEE Communications Magazine, vol. 40, no. 10, pp. 42–51, Oct 2002.

[25] Jéerôme François, Issam Aib, Raouf Boutaba*," FireCol*: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks*",* IEEE 2012 Transaction on Networking, Volume :PP, Issue:99.

[26] Yao Chen1, Shantanu Das1, Pulak Dhar2, Abdulmotaleb El Saddik1, and Amiya Nayak1 School of Information Technology and Engineering, University of Ottawa1, "Detecting and Preventing IP-spoofed Distributed DoS Attacks" International Journal of Network Security, Vol.7, No.1, PP.70–81, July 2008.

