INTERNATIONAL JOURNAL FOR DEVELOPMENT OF COMPUTER SCIENCE & TECHNOLOGY ISSN-2320-7884 (ONLINE)
VOLUME-1, ISSUE-III (April-May 2013) IS NOW AVAILABLE AT: www.ijdcst.com ISSN-2321-0257 (PRINT)

DDoS Counter Measures Based on Snort's detection system

S.Manjari[1], Mr K.Raja Sekhar[2]

[1] Student, K.L.University,Vaddeswaram,Guntur(dt),, Andhra Pradesh, India

[2] Professor, K.L.University,Vaddeswaram,Guntur(dt), Andhra Pradesh, India

ABSTRACT: DDoS attacks are mainly used for flooding a particular victim with massive traffic and paralyzing its services. Recent works aim at countering DDoS attacks by fighting the underlying vector, which is usually the use of botnets. The sudden increase in traffic can cause the server to offer degraded performance. My Doom devastation on Microsoft, wiki leaks encounter with ddos barrages are some examples to highlight the impact. 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 and other major Internet players like Amazon, CNN, and Yahoo are no exception. Early discovery of these attacks, although challenging, is necessary to protect victim server's network infrastructure resources. Previous intrusion prevention systems like FireCol although efficient in thwarting DDoS, its architecture is based on ISP collaboration and virtual protection rings. We propose to use an IPS rules (Snort rules) driven DDoS detection approach that checks various parts of a data packet and not just the header. SNORT is one popular and actively developing open-source Intruder Detection System that uses such a set of signatures known as SNORT rules. This enables the detection system to eliminate other forms DoS attacks such as Slow Read DoS attack. Its effectiveness and low overhead, as well as its support for incremental deployment in real networks are demonstrated.

I INTRODUCTION

DDoS attacks are mainly used for flooding a particular victim with massive traffic and paralyzing its services. Recent works aim at countering DDoS attacks by fighting the underlying vector, which is usually the use of botnets. 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. Unfortunately, detecting a botnet is hard, and an efficient solution requires scanning entities to participate actively in the botnet itself unlike entities scanning from a safe distance. 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. However, even in that latter case, the IDS/IPS may crash because it needs to deal with an overwhelming volume of packets (some flooding attacks reach 10–100 Gb/s). In addition, allowing such huge traffic to transit through the Internet and only detect/block it at the host IDS/IPS may severely strain Internet resources. So a collaborated system is required that can empower the single host based detection and blocking procedures for an efficient prevention of DDoS

Existing technique i.e., 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 ISPs forming overlay networks of protection rings around subscribed customers. The virtual ring uses horizontal communication when the degree of a potential attack is high. In this way, the threat is measured based on the overall traffic bandwidth directed to the customer compared to the maximum bandwidth it supports. FireCol ARCHITECTURE uses the following algorithms: Packet rate computation using rule frequencies (collaboration manager) and Mitigation Sheilds Deployment. In addition to detecting flooding DDoS attacks, FireCol also helps in detecting other flooding scenarios, such as flash crowds, and other botnet-based DDoS attacks thus offering a better performance.

FireCol's defense procedures(virtual protection rings notion) is not based on IPS rule structures(Snort Rules). In this paper, we Extending FireCol to support different IPS rule structures will help FireCol thwart other forms of DoS attacks especillay the latest entrant Slow Read DoS attack. Like viruses, most intruder activity has some sort of signature. Information about these signatures is used to create Snort rules. Snort’s detection system is based on rules. These rules in turn are based on intruder signatures. Snort rules can be used to check various parts of a data packet not just the header scanning adapted by prior approaches. A rule may be used to generate an alert message, log a message, or, in terms of Snort, pass the data packet, i.e., drop it silently. Thus enabling a detection system eliminating other forms 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

IDS

Intrusion Detection System or IDS is software, hardware or combination of both used to detect intruder activity. Snort is an open source IDS available to the general public. An IDS may have different capabilities depending upon how complex and sophisticated the components are. IDS appliances that are a combination of hardware and software are available from many companies.

Network IDS or NIDS

NIDS are intrusion detection systems that capture data packets traveling on the network media (cables, wireless) and match them to a database of signatures. Depending upon whether a packet is matched with an intruder signature, an alert is generated or the packet is logged to a file or database.

Host IDS or HIDS

Host-based intrusion detection systems or HIDS are installed as agents on a host. These intrusion detection systems can look into system and application log files to detect any intruder activity. Some of these systems are reactive, meaning that they inform you only when something has happened. Some HIDS are proactive; they can sniff the network traffic coming to a particular host on which the HIDS is installed and alert you in real time.

Signatures

Signature is the pattern that you look for inside a data packet. A signature is used to detect one or multiple types of attacks. For example, the presence of “scripts/admin” in a packet going to your web server may indicate an intruder activity. Signatures may be present in different parts of a data packet depending upon the nature of the attack. For example, you can find signatures in the IP header, transport layer header (TCP or UDP header) and/or application layer header or payload. Usually IDS depends upon signatures to find out about intruder activity. Some vendor-specific IDS need updates from the vendor to add new signatures when a new type of attack is discovered.

Alerts

Alerts are any sort of user notification of an intruder activity. When an IDS detects an intruder, it has to inform security administrator about this using alerts. Alerts may be in the form of pop-up windows, logging to a console, sending e-mail and so on. Alerts are also stored in log files or databases where they can be viewed later on by security experts.

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. Usually an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it. Snort is primarily a rule-based IDS, however input plug-ins are present to detect anomalies in protocol headers. Snort uses rules stored in text files that can be modified by a text editor. Rules are grouped in categories. 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. Finding signatures and using them in rules is a tricky job, since the more rules use, the more processing power is required to process captured data in real time. It is important to implement as many signatures as it can using 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 ARCHITECTURE OF SNORT

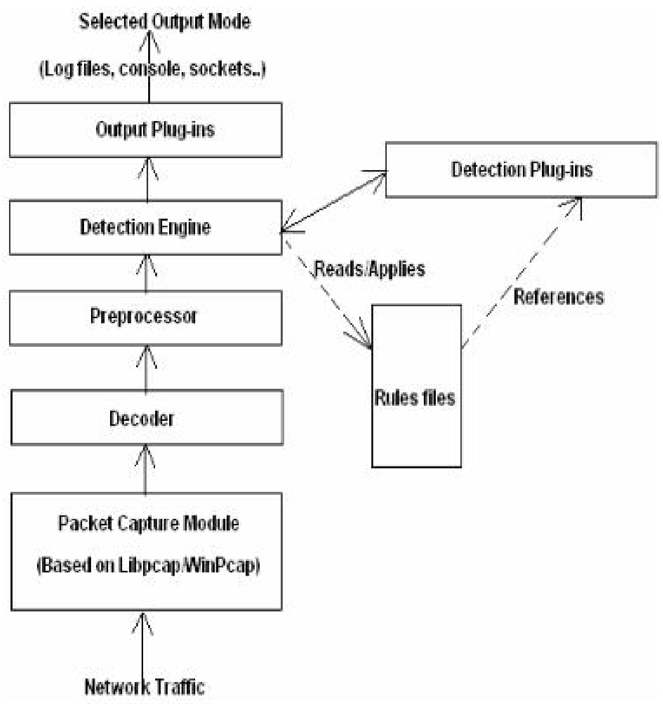

Fig: Snort Architecture

Decoder: It fits captured packets into data structures and identifies link level protocols. Then it takes the next level, decodes IP, and TCP/UDP to get information about port addresses. Snort alerts for malformed headers, unusual TCP option.

Preprocessors: They are like filters, which identifies things that should be checked later in Detection Engine module (like suspicious connection attempt to some TCP/UDP port or too many UDP packets received during a port scan).

Rule files: Text files with rule sets written with a known syntax.

Detection Plug-ins: Those modules referenced from its definition in the rule files, and they are intended to identify patterns whenever a rule is evaluated.

Detection engine: Making use of detection plug-ins, it matches packets against rules previously charged into memory since snort initialization. Output plug-ins: Alerts, logs, extern files, databases.

V PROPOSED SYSTEM

SNORT is one of the most popular NIDS. SNORT is Open Source, which means that the original program source code is available to anyone at no charge, and this has allowed many people to contribute to and analyse the programs construction. SNORT uses the most common open-source licence 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:

Snort's architecture is focused on performance, simplicity, and flexibility. There are three primary subsystems that make up Snort: the packet decoder, the detection engine, and the logging and alerting subsystem. These subsystems ride on top of the libpcap promiscuous packet sniffing library, which provides a portable packet sniffing and filtering capability. Program configuration, rules parsing, and data structure generation takes place before the sniffer section is initialized, keeping the amount of per packet processing to the minimum required to achieve the base program functionality.

(i) The Packet Decoder

The decode engine is organized around the layers of the protocol stack present in the supported data-link and TCP/IP protocol definitions. Each subroutine in the decoder imposes order on the packet data by over-laying data structures on the raw network traffic.

These decoding routines are called in order through the protocol stack, from the data link layer up through the transport layer, finally ending at the application layer. Speed is emphasized in this section, and the majority of the functionality of the decoder consists of
setting pointers into the packet data for later analysis by the detection engine. Snort provides decoding capabilities for Ethernet, SLIP, and raw (PPP) data-link protocols. ATM support is under development.

(ii) The Detection Engine

Snort maintains its detection rules in a two dimensional linked list of what are termed Chain Headers and Chain Options. These are lists of rules that have been condensed down to a list of common attributes in the Chain Headers, with the detection modifier options contained in the Chain Options.

These rule chains are searched recursively for each packet in both directions. The detection engine checks only those chain options which have been set by the rules parser at run-time. The first rule that matches a decoded packet in the detection engine triggers the action specified in the rule definition and returns.

(iii) The Logging/Alerting Subsystem

The alerting and logging subsystem is selected at run-time with command line switches. There are currently three logging and five alerting options. The logging options can be set to log packets in their decoded, human readable format to an IP-based directory structure, or in tcpdump binary format to a single log file. The decoded format logging allows fast analysis of data collected by the system. The tcpdump format is much faster to record to the disk and should be used in instances where high performance is required. Logging can also be turned off completely, leaving alerts enabled for even greater performance improvements.

VI WRITING SNORT RULES

Snort rules are simple to write, yet powerful enough to detect a wide variety of hostile or merely suspicious network traffic. There are three base action directives that Snort can use when a packet matches a specified rule pattern: pass, log, or alert.

Pass rules simply drop the packet. Log rules write the full packet to the logging routine that was user selected at runtime. Alert rules generate an event notification using the method specified by the user at the command line, and then log the full packet using the selected logging mechanism to enable later analysis.

Snort interprets keywords enclosed in parentheses as ``option fields''. Option fields are available for all rule types and may be used to generate complex behaviors from the program. Snort version 1.2.1 has fourteen option fields available:

1. content: Search the packet payload for the a specified pattern.
2. flags: Test the TCP flags for specified settings.
3. ttl: Check the IP header's time-to-live (TTL) field.
4. itype: Match on the ICMP type field.
5. icode: Match on the ICMP code field.
6. minfrag: Set the threshold value for IP fragment size.
7. id: Test the IP header for the specified value.
8. ack: Look for a specific TCP header acknowledgement number.
9. seq: Log for a specific TCP header sequence number.
10. logto: Log packets matching the rule to the specified filename.
11. dsize: Match on the size of the packet payload.
12. offset: Modifier for the content option, sets the offset into the packet payload to begin the content search.
13. depth: Modifier for the content option, sets the number of bytes from the start position to search through.
14. msg: Sets the message to be sent when a packet generates an event.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

These options may be combined in any manner to detect and classify packets of interest. The rule options are processed using a logical AND between them; all of the testing options in a rule must be true in order for the rule to generate a ``found" response and have the program perform the rule action.

VII 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, generalisation 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.

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. The graph showing the bandwidth supported for varying payload sizes is given below:

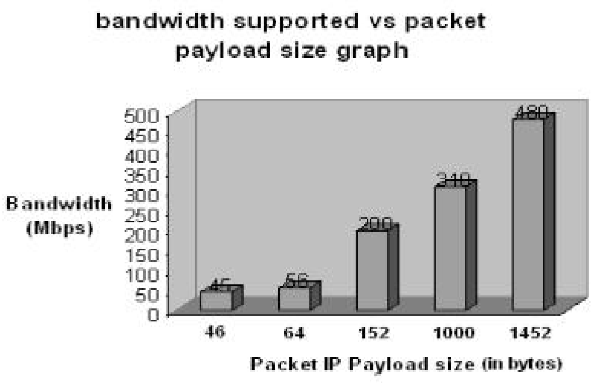

Fig: Bandwidth dependence of packet payload size.

Now Provide the results of Snort's performance when the rule set sizes are varied. We chose to investigate the dependence of bandwidth on the size of the rule set when the Packet size is kept constant. The first case we consider is the scenario when the packet size is 1452 bytes.

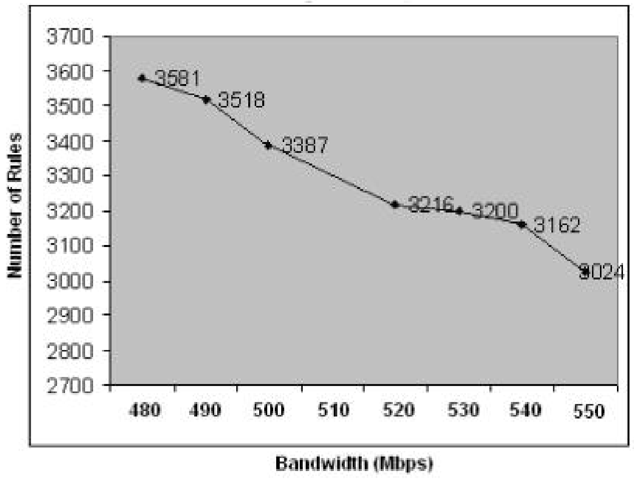

Fig: Dependence of bandwidth supported on rule set size (payload size: 1452 bytes)

In this case, the dependence of the bandwidth supported on the number of rules for packet IP payload size of 46 bytes.

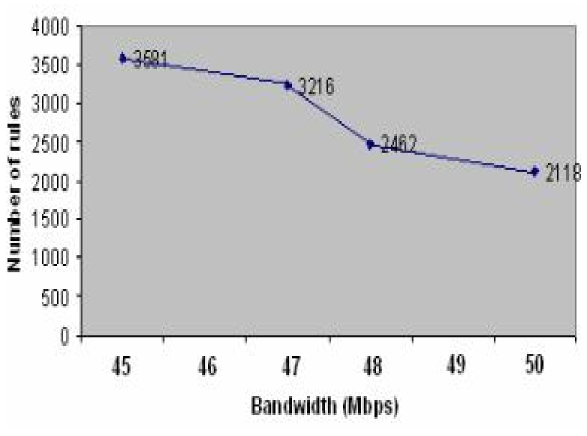

Fig: Dependence of bandwidth supported on rule set size (payload size: 46 bytes)

For a given bandwidth, the number of packets generated will be larger in case of packets with smaller IP payload, than the case where the payload is large. When large numbers of packets are generated, then it becomes difficult to read all of them from the NIC, and therefore packets are dropped

VIII CONCLUSION

In this paper, the proposed system extending FireCol to support different IPS rule structures will help FireCol thwart other forms of DoS attacks especially the latest 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 based detection system consists of several components: Sniffer, preprocessor, the detection engine, the output/ alert component. The detection engines make use of snort rules. Snort rules can be used to check various parts of a data packet not just the header scanning adapted by prior approaches. A rule may be used to generate an alert message, log a message, or, in terms of Snort, pass the data packet, i.e., drop it silently. Thus enabling a detection system eliminating other forms 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.

IX 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