Global Journal of Computer Science and Technology
Volume 11 Issue 21 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Modeling and Counter Measures of Flooding Attacks to Internet Threat Monitors (ITM): Using Botnet and Group-Testing approach

By K Munivara Prasad, A Rama Mohan Reddy, V Jyothsna

Sree Vidyanikethan Engg.College, Tirupati

Abstract - The Internet Threat Monitoring (ITM),is a globally scoped Internet monitoring system whose goal is to measure, detect, characterize, and track threats such as distribute denial of service(DDoS) attacks and worms. To block the monitoring system in the internet the attackers are targeted the ITM system. In this paper we address flooding attack against ITM system in which the attacker attempt to exhaust the network and ITM's resources, such as network bandwidth, computing power, or operating system data structures by sending the malicious traffic. We propose an information-theoretic frame work that models the flooding attacks using Botnet on ITM.we propose a novel group testing (GT)-based approach deployed on back-end servers, which not only offers a theoretical method to obtain short detection delay and low false positive/negative rate, but also provides an underlying framework against general network attacks.

General Terms : *Computer networks, network security, Attacks and Internet.*

Keywords : *Internet Threat Monitors (ITM), DDoS, Flooding attack, Botnet and Honeypot, Group testing.*

GJCST Classification : *K.6.5*

Strictly as per the compliance and regulations of:

Modeling and Counter Measures of Flooding Attacks to Internet Threat Monitors (ITM): Using Botnet and Group-Testing approach

K Munivara Prasad [α], A Rama Mohan Reddy [Ω], V Jyothsna [β]

Abstract - The Internet Threat Monitoring (ITM),is a globally scoped Internet monitoring system whose goal is to measure, detect, characterize, and track threats such as distribute denial of service(DDoS) attacks and worms. To block the monitoring system in the internet the attackers are targeted the ITM system. In this paper we address flooding attack against ITM system in which the attacker attempt to exhaust the network and ITM's resources, such as network bandwidth, computing power, or operating system data structures by sending the malicious traffic. We propose an information-theoretic frame work that models the flooding attacks using Botnet on ITM.we propose a novel group testing (GT)-based approach deployed on back-end servers, which not only offers a theoretical method to obtain short detection delay and low false positive/negative rate, but also provides an underlying framework against general network attacks.

General Terms : Computer networks, network security, Attacks and Internet.

Keywords : Internet Threat Monitors (ITM), DDoS, Flooding attack, Botnet and Honeypot, Group testing.

I. Introduction

Internet security is increasing in importance. Yet, despite decades of research, we are still unable to make secure computer networks. Further, more sophisticated and new attacks are expected to continue posing a greater degree of threat to Internet services. As a result, a more fundamental model, in terms of theoretical and system perspectives, regardless of attack types must be investigated. An essential problem to overcome for any defense mechanism is the fact that malicious traffic/packets can be similar to legitimate ones.

Denial-of-Service (DoS) is a major security problem in computer systems and networks. In a DoS attack, a group of attackers try to make a service unavailable to legitimate clients for unacceptably long periods of time. Service-level DoS attacks target server resources by issuing legitimate-like service requests at a high rate to overwhelm the victim servers. These attacks, attempt to exhaust the victim's resources, such as network bandwidth, computing power, or operating system data structures. Flood attack, Ping of Death attack, SYN attack, Teardrop attack, DDoS , and Smurf attack are the most common types of DoS attacks. The hackers who launch DDoS attacks typically target sites or services provided by high-profile organizations, such as government agencies, banks, credit-card payment gateways, and even root name servers.

A flooding-based Distributed Denial of Service (DDoS) attack is a very common way to attack a victim machine by sending a large amount of unwanted traffic. Network level congestion control can throttle peak traffic to protect the network. Network monitors are used to monitor the traffic in the networks to classify them as genuine or attack traffic and also these monitors gives the traffic as an input to several DDoS detection algorithms for detection of DDoS attacks. However, it cannot stop the quality of service (QoS) for legitimate traffic from going down because of attacks. Two features of DDoS attacks hinder the advancement of defense techniques. First, it is hard to distinguish between DDoS attack traffic and normal traffic. There is a lack of an effective differentiation mechanism that results in minimal collateral damage for legitimate traffic. Second, the sources of DDoS attacks are also difficult to find in a distributed environment. Therefore, it is difficult to stop a DDoS attack effectively.

The Internet Threat Monitoring (ITM) System basically has two main components one is centralized data center and another is the number of monitors which are distributed across the Internet. Each monitor covers the range of IP addresses and monitors the traffic to send the traffic logs to data center. The data center now collects the traffic logs from monitors and analyzes the collected traffic logs to publish reports to ITM system users.

The collected logs, as a random sample of the Internet traffic, can still provide critical insights for the public to measure, characterize, and track/detect Internet security threats. The idea of ITM systems dates back to DShield and CAIDA network telescope [4], [5], which have been successfully used to analyze the activities of worms and DDoS attacks [3], [6].The reason is that if an attacker discovers the monitor locations, it can easily avoid detection (by ITM systems) by bypassing the monitored IP addresses and directing the attack to the much larger space of unmonitored IP addresses. Furthermore, such an attacker may even mislead the reports published by an ITM system by

Author [α]: Assistant Professor(SL) Sree Vidyanikethan Engg.College, Tirupati. E-mail : prasadkmv27@gmail.com

Author [Ω] : Professor and Head Dept.of CSE Sri Venkateswara university college of Engg.S V University,Tirupati.

Author [β] : Assistant Professor, Department of Information technology, Sreevidyanikethan Engineering College, Tirupati

manipulating traffic to the identified monitors, generating highly skewed samples. Since ITM reports are trusted by the public as a random (unbiased) sample of Internet traffic, the confidentiality of monitor locations is vital for the usability of ITM systems.

The monitor locations of an ITM system can be compromised by introducing several attacks by the attackers which includes Localization attacks [1] and DDoS Attacks which exploits some vulnerability or implementation bug in the software implementation of a service to bring that down or that use up all the available resources at the target machine or that consume all the bandwidth available to the victim machine, this is called as Bandwidth attacks.

The main goal of our work is to perform the identification of attackers much faster by testing them in group instead of one by one. This will help to detect the flooding attacks, So that if any ITM is found under attack, we can immediately identify and filter the attackers out of its client set. Apparently, this problem resembles the group testing (GT) theory [14] which aims to discover defective items in a large population with the minimum number of tests where each test is applied to a subset of items, called pools, instead of testing them one by one. Therefore, we apply GT theory to this network security issue and propose specific algorithms and protocols to achieve high detection performance in terms of short detection latency and low false positive/negative rate.

In this paper we introduce an information theoretic frame work model to existing flooding attacks on ITM system monitors. In the flooding attack the attacker sends the large volume of unwanted traffic to the targeted monitor by using the botnet or huge number of compromised systems. Based on the Information-theoretic model we propose a Group Testing based approach to detect flooding attacks.

II. Related Work

Probing traffic based Localization attack [7][8] in which an attacker sends high rate short length port scan messages to the targeted network to compromise the monitor locations in ITM system. Then, attacker queries the data center to determine whether a short spike of high-rate traffic appears in the queried time-series data, for confirmation of the attack.

A steganographic localization attack [9] an attacker launches a stream of low-rate port-scan probing traffic which is marginally modulated by a secret Pseudonoise (PN) code. While the low-rate property prevents the exhibition of obvious regularity of the published traffic data at the data center, based on the carefully synchronized PN code, the attacker can still accurately identify the PN-code-modulated traffic in the retrieved published traffic data from the data center. Thereby, the existence of monitors in the targeted network can be compromised. To this end, the PN-code-based steganographic attack presented in our paper can be understood as a covert channel problem [10], because the attack traffic encoded by a signal blends into the background traffic and is only recognizable by the attacker which knows the secret pattern of the PN code.

In [1] introduced the information theoretic framework to evaluate the effectiveness of the localization attacks by using the minimum time length required by an attacker to achieve a predefined detection rate as the metric. But this frame work is defined in specific to the localization attacks only; they are not given any solution for other DDoS attacks. The frame work allows the ITMs which are registered within the data center given, and the access is restricted to that private region only. But public access of the ITMs and data center allows more scope to provide security against different attacks.

Group Testing (GT) based approach [17] is used to detect the application denial of service attacks, there the efficiency of the attack detection is improved by testing the traffic by group instead of one by one. The GT approach also minimizes the false positives and false negatives comparatively to the input traffic. But the GT approach can be applied to the DDoS detection whenever the number of attackers known in advance. This assumption will not be suitable for all DDoS attacks.

III. Proposed Work

In [1] the authors define a model in which the ITMs in the networks sends the traffic logs periodically to the data center and the data center collects the traffic logs and publishes the reports to ITM system users which are registered, that means it creates the private environment or region .In the private region the scope for DDoS attacks are very less, and they are restricted this model only for Localization attacks. In this section we have defined a model which will provide the following extensions.

Public accessing : Public accessing of the data center increases the network usage and provides better communication with the outside world rather than private environment. In this any user from outside the private region can get the communication with the private network, if the user is genuine he can get the status of the monitor before sending the data to internal monitors, to avoid the attacks. If the user is an attacker, then this status information can be misused to perform the attacks on the monitor. The data center sends the status information to any users (public or private) based on the request query, but the private (internal) users can get the highest priority.

Usage of Botnets for Flooding Attack : A denial-of-service (DoS) attack is an explicit attempt by attackers to prevent an information service's legitimate users from using that service. In a DDoS attack, these

attempts come from a large number of distributed hosts that coordinate to flood the victim with an abundance of attack packets simultaneously. The attacker may use the botnets [11], [12] and other alternatives to launch the attack.

a) Flooding

Launching a flooding attack : Once the DDoS network has been set up and the infrastructure for communication between the agents and the handlers established, all that an attacker needs to do is to issue commands to the agents to start sending packets to the victim host. The agents try to send unusual data packets (all TCP flags set, repeated TCP SYN packets, Large ICMP packets) to maximize the possibility of causing disruption at the victim and the intermediate nodes. There are certain basic packet attack types which are favorites of the attack tool designers. All the attack tools use a combination of these packet attack types to launch a DDoS attack. The basic attack types are

i. TCP floods : A stream of packets with various flags (SYN,RST, ACK) are sent to the victim machine. The TCP SYN flood works by exhausting the TCP connection queue of the host and thus denying legitimate connection requests. TCP ACK floods can cause disruption at the nodes corresponding to the host addresses of the floods as well. Also the one known tool that uses TCP ACK flooding (mstream [13]) has been known to cause disruptions in a router even with a moderate packet rate. Both TCP SYN flooding and the mstream attack constitute a group of attacks known as asymmetric attacks (Attacks where a less powerful system can render a much more powerful system useless).

ii. ICMP floods (e.g ping floods) : A stream of ICMP packets is sent to the victim host. A variant of the ICMP floods is the Smurf attack in which a spoofed IP packet consisting of an ICMP ECHO_REQUEST is sent to a directed broadcast address. The RFC for ICMP specifies that no ECHO_REPLY packets should be generated for broadcast addresses, but unfortunately many operating systems and router vendors have failed to incorporate this into their implementations. As a result, the victim host (in this case the machine whose IP address was spoofed by the attacker) receives ICMP ECHO_REPLY packets from all the hosts on the network and can easily crash under such loads. Such networks are known as amplifier networks and thousands of such networks have been documented.

iii. UDP floods : A huge amount of UDP packets are sent to the victim host. Trinoo is a popular DDoS tool that uses UDP floods as one of its attack payloads.

b) Bots

Studying the evolution of bots and botnets provides insight into their current capabilities. One of the original uses of computer bots was to assist in Internet Relay Chat (IRC) channel management [14]. IRC is a chat system that provides one-to-one and one-to-many instant messaging over the Internet. Users can join a named channel on an IRC network and communicate with groups of other users. Administering busy chat channels can be time consuming, and so channel operators created bots to help manage the operation of popular channels. One of the first bots was Eggdrop, which was written in 1993 to assist channel operators [1].

In time, IRC bots with more nefarious purposes emerged. The goal of these bots was to attack other IRC users and IRC servers. These attacks often involved flooding the target with packets (i.e., DoS attacks). The use of bots helped to hide the attacker because the attack packets were sent from the bot rather than directly from the attacker (assuming a non-spoofed attack). This new level of indirection also allowed multiple computers to be grouped together to perform distributed attacks (DDoS) and bring down bigger targets. Larger targets required more bots, and so attackers looked for methods to recruit new members. Since very few users would agree to have their computers utilized for conducting packet floods, attackers used trojaned files and other surreptitious methods to infect other computers.

c) IRC- based Command and Control

A bot must communicate with a controller to receive commands or send back information. One method for establishing a communication channel is to connect directly to the controller. The problem is that this connection could compromise the controller's location. Instead, the bot controller can use a proxy such as public message drop point (e.g., a well-known message board). However, because websites and other drop points can introduce significant communication latency, a more active approach is desirable. A well-known public exchange point that enables virtually instant communication is IRC.

IRC provides a common protocol that is widely deployed across the Internet and has simple text-based command syntax. There is also a large number of existing IRC networks that can be used as public exchange points. In addition, most IRC networks lack any strong authentication, and a number of tools to provide anonymity on IRC networks are available. Thus, IRC provides a simple, low-latency, widely available, and anonymous command and control channel for botnet communication. An IRC network is composed of one or more IRC servers as depicted in Figure 1.

In a typical botnet, each bot connects to a public IRC network or a hidden IRC server on another compromised system. The bot then enters a named channel and can receive commands directly from a controller or even from sequences encoded into the title of the channel.

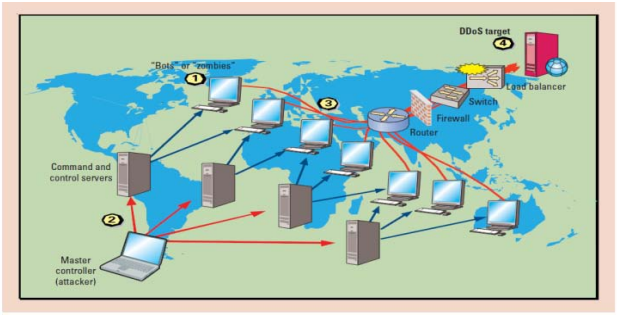

Figure 1 : Compromised computers. In a distributed denial-of-service attack (DDoS), these computers serve three major roles: master controller, command and control server, and bot.

d) Group Testing

The first application of group testing was during WWII; instead of testing every blood sample individually, groups of samples were pooled together and tested collectively. If the outcome of the group test is negative, all samples in the group are good (disease-free). Although group testing has been used since then in many security and networking applications, such as data forensics, cryptography, multiple-access channels, and broadcast security against jamming, our work is the first to apply this powerful theory to the DoS attack problem.

Group testing aims mainly at identifying the defective (special) members of a population with few tests. There are two classes of group-testing mechanisms. "Non-adaptive", or single-stage, specifies all tests simultaneously without the benefit of using the outcomes of previous tests to determine the present test. Adaptive (multi-stage) group testing uses feedback from previous test results to determine subsequent tests.

i. *Basic Idea*

The basic group testing is modeled as a T × N matrix, where N is the total number of members, and T is the number of tests. Matrix rows represent tests and columns represent group members. When a matrix element (i, j) is set to 1, this means that member j participates in test i. An example of a group testing matrix for a population of 10 members with 4 tests is shown in Figure 1. Test results are represented as a vector with an element for each test. For simplicity we assume that the test results are binary. So, a test result is set to 1 if the corresponding test returns a positive result, that is, if the test was applied to a group with at least one defective member. In the example shown in Figure 1 the 2nd, 6th, and 8th members are defective.

	Members (defectives underlined)										*Test Results*
	1	2	3	4	5	6	7	8	9	10	
	0	1	0	0	0	0	1	1	1	0	1
	1	0	1	0	0	0	1	0	1	1	0
Tests	0	0	0	1	0	1	0	1	0	1	1
	1	0	0	1	1	0	1	0	0	0	0

ii. *Detection of defective members*

The detection algorithm discovers the defective members using the result vector and Members Test (defectives underlined) Results. An example of a group-testing matrix. the matrix. The algorithm we use in this paper works by excluding a negative (non-defective) member if it participates in a "large enough" number of tests with a negative result. For instance, if we assume that a member has to participate in only one negative test to be excluded, then in the above example; members 1, 3, 4, 5, 7, 9, and 10 will be excluded. This leaves us with the defective members 2, 6, and 8 as suspects. In this specific example, all defective elements are detected, and all non-defective members are cleared.

iii. *Apply to Attack Detection*

A detection model based on GT can be assumed that there are T virtual servers and N clients, among which d clients are attackers shown in fig. Consider the matrix M_{txn} ,the clients can be mapped into the columns and virtual servers into rows in M, where M [i,j]=1 if and only if the requests from client j are distributed to virtual server i. With regard to the test outcome column V, we have V [i]=1 if and only if virtual server i has received malicious requests from at least one attacker, but we cannot identify the attackers at once unless this virtual server is handling only one client. Otherwise, if V [i]=0, all the clients assigned to server i are legitimate. The d attackers can then be captured by decoding the test outcome vector V and the matrix M.

iv. *False Positive and False Negative Probabilities*

A false positive is when a non-defective member gets falsely identified as defective, while a false negative is when a defective member ends up being not detected. In the example above, both the false positive probability and the false negative probability are 0. In general, simple detection algorithm discussed above detects all defective members with the false positive probability

$$FP = [1-p(1-p)d]T,$$

Where d is the number of defective members in the group and T is the number of tests used to detect defective members. By differentiating the above equation with respect to p, the optimal value of p, the value that yields the minimum false positive probability, is 1/d+1. Thus, the minimum false positive probability for a given number of tests T is:

$$FP = [1 - \frac{1}{d+1}(1 - \frac{1}{d+1})^d]^T$$

Finally, from Eqn. 1 we derive the number of tests, Tfp, required to achieve a target false positive probability fp:

$$T_{fp} = \frac{log(fp)}{log(1 - \frac{1}{d+1}(1 - \frac{1}{d+1})^d)}$$

As we will show later, Tfp is O(d), that is, the number of tests is in the order of number of defective members attackers) not the total number of members (N).

The false negative probability is

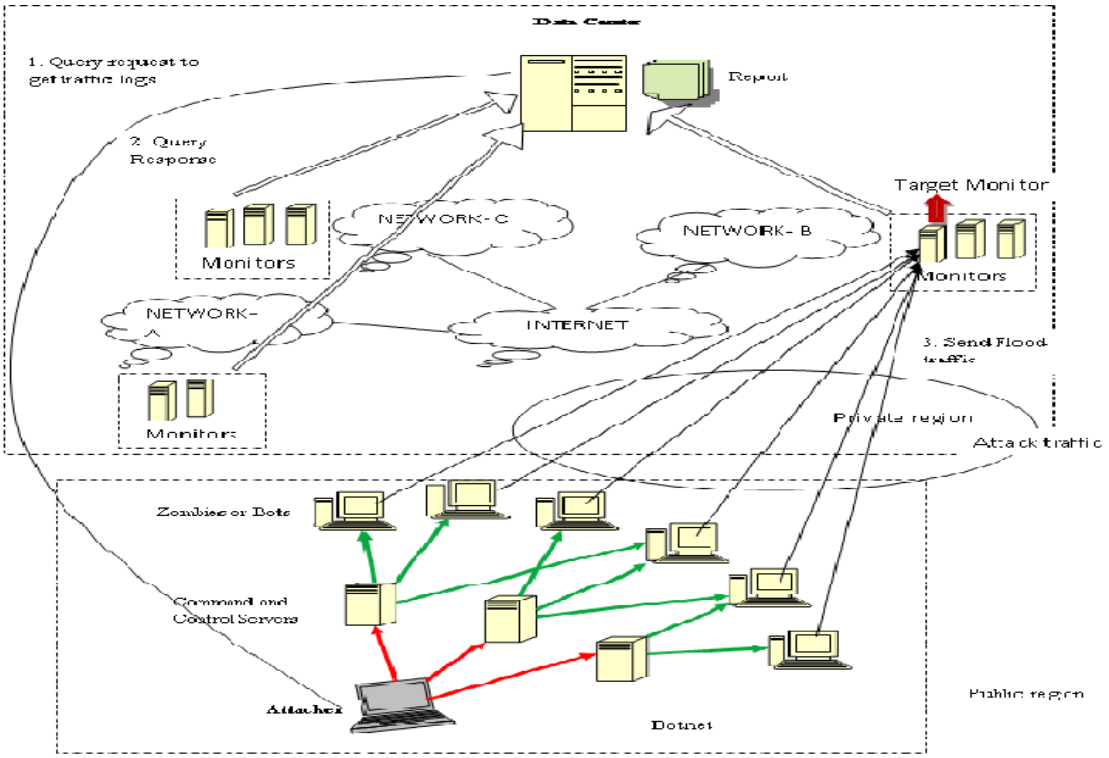

Figure 2 : Work flow of flooding attacks using botnet.

$$FN = 1 - [1 - \frac{1}{d+1}(1.0 - \rho_{attack})(1 - \frac{1}{d+1} \cdot \rho_{attack})^{d-1}]^T$$

Where ρattack refers the the probability of attack and d is the expected number of attackers.

IV. Proposed Model

In this paper we divided the entire model into two regions namely private region and public region. The Internet Threat Monitors (ITM) are distributed across the Internet and each monitor records the traffic addressed to range of IP addresses and send the traffic logs periodically to the data center. The data center then analyzes the traffic logs collected from the monitors and publishes the reports to ITM system users. The collection of monitors under the data center forms the private region because the ITMs are registered before sending the logs to the data center. Any user can get the reports of the requested ITM by sending the query request to the data center and the data center is answerable to all the ITMs which are registered.

The public region of our model specifies the unregistered users of the data center who does not have any permission to access the data center, but they can get the traffic reports related to any ITM by sending the query request to the data center. The data center scope is extended to the public domain but it can only give the traffic reports to the public users. Allowing the public users or network accessing to the data center and monitors, causes decrease in the performance because of the overload of the data center. These can be balanced by introducing the priorities to the users; the internal or private region users have the highest priority than the public users .This priorities does not disturb the existing scenario but this can enhance the service to the public domain ,this will not be a over burden to the data center.

In This section we are constructing the botnet as the public user network without having any registration with data center and performing the flooding attack on the ITM which is local to the data center.

i. *Generation of flooding attack with Botnet*

A DDoS (Flooding) attack mechanism typically includes a network of several compromised computers [15]. These compromised computers serve three major role -master controller, command and control (C&C) server, and bot. An attacker prepares a DDoS attack by exploiting vulnerabilities in one computer system and making it the DDoS "master controller." From here, the attacker identifies and communicates with other

compromised systems. A C&C server is a compromised host with a special program running on it, this server distributes instructions from the attacker to the rest of the bots, which form a botnet[11]. (A bot is a compromised host that runs a special program.) Each C&C server is capable of controlling multiple bots, each of which is responsible for generating a stream of packets to the intended victim. Often, the bots employed to send the flood of requests are infected with a virus that lets attackers use them anonymously.

A Flooding attack happens in several phases :

- Discover vulnerable hosts. To launch a DDoS attack, attackers first build a network of computers that they can use to produce the volume of traffic needed to deny services to legitimate users. To create this network, they first scan and identify vulnerable sites or hosts. Vulnerable hosts are usually those that run either no antivirus software or an out-of-date version, or those that aren't properly patched. Attackers use these compromised hosts for further scanning and compromises.
- Establish a botnet. After gaining access, attacker must then install attack tools on the compromised hosts to form a botnet.
- Launch an attack. In the next phase, attackers send commands to C&C servers for their bots to attack by sending hundreds of thousands of requests to the target simultaneously.
- Flood a target. In the final phase, monitor receives a flood of requests to the point where they can't operate effectively.

ii. *Conformation of attack*

The attacker queries the data center for the traffic reports. Such traffic reflects both flooding requests traffic and other traffic collected from all monitors. Then the attacker confirms the attack by checking the status of the ITM in the traffic reports published by the data center.

V. Prevention

Preventive mechanisms attempt either to reduce the possibility of DDoS attacks or enable potential victims to endure the attack without denying services to legitimate users.

- System security mechanisms increase a host's overall security posture and prevent it from becoming part of a botnet or a DDoS victim. Examples of system security mechanisms include reliable firewall filtering, proper system configuration, effective vulnerability management, timely patch installation, robust antivirus programs, controlled and monitored system access, and solid instruction detection.
- Resource multiplication mechanisms provide an abundance of resources to counter DDoS threats, such as increasing the capacity of network bandwidth, routers, firewalls, and servers. Additional examples include deploying information services at diverse network locations and establishing clusters of servers with load-balancing capabilities. Resource multiplication essentially raises the bar on how many bots must participate in an attack to be effective. While not providing perfect protection, this last approach has often proved sufficient for small- to mid-range DDoS attacks.

Preventing Flooding Attacks

In this section we introduce a general methodology to prevent flooding attacks. It is based on the following line of reasoning:

1) To mount a successful Flooding attack, a large number of compromised machines are necessary.
2) To coordinate a large number of machines, the attacker needs a remote control mechanism.
3) If the remote control mechanism is disabled, the Flooding attack is prevented.

Our methodology to mitigate flooding attacks aims at manipulating the root-cause of the attacks, i.e., influencing the remote control network. Our approach is based on three steps:

1. Infiltrating the remote control network.
2. Analyzing the network in detail.
3. Shutting down the remote control network.

In the first step, we have to find a way to smuggle an agent into the control network. In this context, the term agent describes a general procedure to mask as a valid member of the control network. This agent must thus be customized to the type of network we want to plant it in. The level of adaptation to a real member of the network depends on the target we want to infiltrate. For instance, to infiltrate a botnet we would try to simulate a valid bot, maybe even emulating some bot commands.

Once we are able to sneak an agent into the remote control network, it enables us to perform the second step, i.e., to observe the network in detail. So we can start to monitor all activity and analyze all information we have collected.

In the last step, we use the collected information to shut down the remote control network. Once this is done, we have deprived the attacker's control over the other machines and thus efficiently stopped the threat of a flooding attack with this network. Again, the particular way in which the network is shut down depends on the type of network.

VI. Detection of Flooding Attacks

In this section we present efficient way of detecting the attacks on the ITMs in the given information theoretic frame work. We divide the attack

detection process into two phases, Firstly the primary detection of DDoS attacks on the ITMs and the later is the detection of flooding attacks on the ITMs.

In the primary detection phases the system detects the attacks based on traffic information aggregated from all monitors in the ITM system. If the overall traffic rate (e.g., volume in a given time interval) exceeds a predetermined threshold, the defender issues an alarm. The threshold value can be maintained either at data center or the individual ITMs based on the type of schemes used [1] in the network.In the primary detection phase the system detects some attack was happened in the network. If the detection scheme is centralized, then whenever the aggregate traffic exceeds the threshold maintained at the data center then the data center finds the attack and that attacked monitor can be identified by verifying the individual traffic logs of each ITM from the report. Otherwise if the detection strategy is distributed then each monitor maintained an individual threshold and checked the aggregate traffic regularly. If the traffic exceeds the threshold then it find the attack was happened and sends the status as attacked to the data center. After getting the attacked status from the ITM the data center blocks the corresponding ITM and displays the status of the ITM as blocked in the status reports, which will avoids the further traffic to or from the attacked ITM with the rest of the networks.

The second stage of detection specifies the detection of the flooding attacks. Once the attack is conformed then the data center identifies the attacked monitor and the traffic logs will be handover to the flooding detection phase. The flooding detection phase then performs the group testing (GT) on the traffic, then identifies the attackers from the client traffic set.

In this section we define group testing (GT) based DDoS detection methods for flooding detection on ITMs in information frame work defined, and also detection of false positives and false negatives in the network for large flow size.

a) BOTNET Detection

Botnets are a very real and quickly evolving problem that is still not well understood. In this paper, we outline the problem and investigate methods of stopping bots. We identify three approaches for handling botnets:

1) Prevent systems from being infected,
2) Directly detect command and control communication among bots and between bots and controllers, and,
3) Detect the secondary features of a bot infection such as propagation or attacks.

The first approach is to prevent systems from being infected. There are a range of existing techniques, including anti-virus software, firewalls, and automatic patching.

The second approach is to directly detect botnet command and control traffic. Botnets today are often controlled using Internet Relay Chat (IRC) and one possible method of detecting IRC-based botnets is to monitor TCP port 6667 which is the standard port used for IRC traffic. One could also look for non-human behavioral characteristics in traffic, or even build IRC server scanners to identify potential botnets.

We argue there is also a third approach that detects botnets by identifying secondary features of a bot infection such as propagation or attack behavior. Rather than directly attempting to find command and control traffic, the key to this approach is the correlation of data from different sources to locate bots and discover command and control connections.

In this paper we investigate the second approach for stopping botnets. The problem with the first approach is that preventing all systems on the Internet from being infected is nearly an impossible challenge. As a result, there will be large pools of vulnerable systems connected to the Internet for many years to come.

b) Detecting Command and Control

To combat the growing problem of bots, we identified two approaches for detecting botnets: detect the command and control communication, or detect the secondary features of a bot infection. In this section we study methods of detecting botnets by directly locating command and control traffic.

i. *IRC-based Botnet Detection*

Today, most known bots use IRC as a communication protocol, and there are several characteristics of IRC that can be leveraged to detect bots. One of the simplest methods of detecting IRC-based botnets is to offramp traffic from a live network on known IRC ports (e.g., TCP port 6667) and then inspects the payloads for strings that match known botnet commands. Unfortunately, botnets can run on non-standard ports. Another method is to look for behavioral characteristics of bots. One study found that bots on IRC were idle most of the time and would respond faster than a human upon receiving a command. The system they designed looked for these characteristics in Netflow traffic and attempted to tag certain connections as potential bots [15].

The approach was successful in detecting idle IRC activity but suffered from a high false positive rate. Given problems such as false positives on live networks, another approach is to use a non-productive resource or honeypot.

One group set up a vulnerable system and waited for it to be infected with a bot. They then located outgoing connections to IRC networks and used their own bot to connect back and profile the IRC server [16]. However, they did not take the next step and develop a detection system based on the technique.

Rather than connecting to the IRC server directly, another approach is to use a honeypot to catch the bot and then look for characteristics of command and control traffic in the outgoing connections. We located all successful outgoing TCP connections and verified that they were all directly related to command and control activity by inspecting the payloads. There were a wide range of interesting behaviors, including connections from the bot to search engines to locate and use bandwidth testers, downloading posts from popular message boards to get server addresses, and the transmission of comprehensive host profiles to other servers.

These profiles included detailed information on the operating system, host bandwidth, users, passwords, file shares, filenames and permissions for all files, and a number of other minute details about the infected host.

We then analyzed all successful outgoing connections and looked for specific characteristics that could be used to identify botnet command and control traffic. The results suggested that there are no simple characteristics of the communication channels themselves that can be used for detection. For example, the length of the outgoing connections varied widely, with certain connections lasting more then 9 hours and others less than a second.

The number of bytes transferred per connection also varied widely even when we separated out IRC communication from other command and control activity. The results from our analysis nor the results from previous bot detection efforts has revealed any simple connection-based invariants useful for network detection. One might inspect every payload of every packet however this is currently very costly on high throughput networks. More importantly, attackers can make small modifications that make detection nearly impossible.

ii. *Limitations of Honeypot detection*

Efficiency : The efficiency of the detection system is depends on the number of honeypots placed in the network. If one honeypot is used to perform the detection in centralized approach, then more than one ITM is attacked automatically the honeypot will be overloaded and it takes more time to detect the attackers. Otherwise if the detection system is distributed, then the efficiency of the detection system is improved, but it is very much cost effective, practically not possible for the large networks.

In GT approach the detection process can be carried out only at data center by collecting the traffic logs from the attacked ITM through data center same as the centralized detection of the honeypot approach. Unlike honeypot detection, the efficiency of the detection process does not depend on the traffic because huge amount of traffic also processed in terms of pools in the GT approach and handled successfully. The pools or groups can be as inputs for multiple rounds of different tests in GT approach to check for different anomalies in the input malicious traffic.

Reliability : One honeypot for each ITM approach is reliable but it is practically very difficult to manage and maintained. If the centralized honeypot compromises then total detection process will be vanished.

The reliability of the GT approach depends on the groups or pools of the malicious traffic considered as the input for the GT approach. The number of tests performed on the traffic improves the detection efficiency and covers wide range of possible attacks of DDoS attacks on ITM.

Scalability : If the detection approach is centralized then no need to use additional honeypots to the network except the honeypot placed at the data center when ever new ITM entered into the private region. In the distributed detection approach new honeypot is attached whenever the new ITM entered into the network.

When the network increases or new ITMs entered into the private region of the network, it does not create additional load on the existing data center. The GT approach handles the input traffic without depends on the number of ITMs or the data centers in the network.

Load Sharing : Every honeypot has its own capacity of handling the load or traffic in the network. In the centralized detection approach if the traffic exceeds its capacity, then the total detection system is vanished. The same problem occurs in case of distributed detection approach also.

If the load in the network increases then the GT approach forms more number of input pools and the tests are applied on the pools repeatedly to perform the attack detection.

False positives and false negatives: In the honeypot based flooding detection false positives and false negatives are not explicitly considered. The detection process finds the root of the attack, by blocking the IRC server.

In GT approach the false positives and false negatives calculated explicitly by conducting specific tests on the input pools of the traffic. False positives and false negatives improve the detection process in terms of considering the attack traffic as genuine and vice-versa.

c) Attack detection using Group testing approach

In the detection model[17], each testing pool is mapped to a virtual server within a back-end server machine. Although the maximum number of virtual servers can be extremely huge, since each virtual server requires enough service resources to manage client requests, it is practical to have the virtual server quantity

(maximum number of servers) and capacity (maximum number of clients that can be handled in parallel) constrained by two input parameters K and w, respectively.

The maximum number of attackers d is assumed known beforehand. Scenarios with nondeterministic d are out of the scope of this paper. In fact, these scenarios can be readily handled by first testing with an estimated d, then increasing d if exactly d positive items are found.

Each back-end server works as an independent testing domain, where all virtual servers within it serve as testing pools. In the following sections, we only discuss the operations within one backend server, and it is similar in any other servers. The detection consists of multiple testing rounds, and each round can be sketched in four stages.

First, generate and update matrix M for testing.

Second, "assign" clients to virtual servers based on M. The back-end server maps each client into one distinct column in M and distributes an encrypted token queue to it. Each token in the token queue corresponds to a 1-entry in the mapped column, i.e., client j receives a token with destination virtual server i iff M[i,j]= 1. Being piggybacked with one token, each request is forwarded to a virtual server by the virtual switch. In addition, requests are validated on arriving at the physical servers for faked tokens or identified malice ID. This procedure ensures that all the client requests are distributed exactly as how the matrix M regulates and prevents any attackers from accessing the virtual servers other than the ones assigned to them.

Third, all the servers are monitored for their service resource usage periodically, specifically, the arriving request aggregate (the total number of incoming requests) and average response time of each virtual server are recorded and compared with some dynamic thresholds to be shown later. All virtual servers are associated with positive or negative outcomes accordingly.

Fourth, decode these outcomes and identify legitimate or malicious IDs. By following the detection algorithms all the attackers can be identified within several testing rounds. To lower the overhead and delay introduced by the mapping and piggybacking for each request, the system is exempted from this procedure in normal service state. As shown in Fig. 3, the back-end server cycles between two states, which we refer as NORMAL mode and DANGER mode. Once the estimated response time (ERT) of any virtual server exceeds some profile-based threshold, the whole backend server will transfer to the DANGER mode and execute the detection scheme. Whenever the average response time (ART) of each virtual server falls below the threshold, the physical server returns to NORMAL mode.

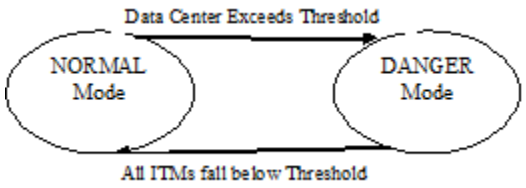

Fig.3 : Two state Diagram if the system.

Based on the system framework above, we propose three detection algorithms SDP, SDoP, and PND in this section. Note that the length of each testing round is a predefined constant P; hence, we analyze the algorithm complexity in terms of the number of testing rounds for simplicity.

i. *Sequential Detection with Packing*

This algorithm investigates the benefit of classic sequential group testing, i.e., optimizing the grouping of the subsequent tests by analyzing existing outcomes. Similar to traditional sequential testing, each client (column) only appears in one testing pool (server) at a time. However, to make full use of the available K servers, we have all servers conduct test in parallel.

ii. *Sequential Detection without Packing*

Considering the potential overload problem arises from the "packing" scheme adopted in SDP, we propose another ralgorithm where legitimate clients do not shift to other servers after they are identified. This emerges from the observation that legitimate clients cannot affect the test outcomes since they are negative.

The basic idea of the SDoP algorithm can be sketched below. Given a suspect IDs set S with initial size n, evenly assign them to the K server machines, similar to SDP in the first round. For the following rounds, assign suspect IDs to the K servers instead of |A| available ones. For the identified legitimate IDs, never move them until their servers are to be overloaded. In this case, reassign all legitimate IDs over the K machines to balance the load. For server with positive outcome, the IDs active on this server but not included by the set of identified legitimate ones, i.e., suspect IDs, will still be identified as suspect. However, if there is only one suspect IDs of this kind in a positive server, this ID is certainly an attacker.

iii. *Partial Non adaptive Detection*

Considering the fact that in the two sequential algorithms mentioned, we cannot identify any attackers until we isolate each of them to a virtual server with negative outcome, which may bring up the detection latency. In this scenario, the requests from the same client will be received and responded by different servers in a round-robin manner. Different from SDP and SDoP, a d-disjunct matrix is used as the testing matrix in this scheme and attackers can be identified without the need of isolating them into servers.

The attack traffic can be identified by using any of the three methods defined and QoS of the system is

completely depends on the number of tests performed, number of rounds conducted on the pools .Once the attack traffic is identified ,then it is easy to find the attack source using the traffic entities. In our paper we focused on the flooding based attacks and these are generated using the botnet. The attacker floods the target ITM by sending the commands through C & C server. Here the attack traffic contains the C & C server address, and then it is very easy to block or point out that server to avoid the flooding attacks on the system.

While identifying the attack traffic with GT approach one can remember that the data should not be lost; these can be effectively done with false positives and negatives.

Despite the number of needed testing rounds differs for these three algorithms above, the time complexity of calculating each testing round for these algorithms is approximate in practice. It is trivial to see that the costs for SDP and SDoP are negligible, but not for PND algorithm which involves polynomial computation on Galois Field. However, considering that the upper bound of both the number of clients n and attackers d is estimated, the detection system can pre compute the d-disjunct matrices for all possible (n, d) pairs offline, and fetch the results in real time. Therefore, the overhead can be decreased to O(1) and the client requests can be smoothly distributed at the turn of testing rounds without suffering from long delays of matrix update.

VII. Conclusion and Future Work

The frame work integrates active real time flooding attack flow identification from botnet with GT approach. The GT approach has been used at Data center to detect the attack traffic that interns helpful, while identifying the C&C server to block the flooding attack against the ITM. The false positive and false negatives can be effectively minimized to improve the QoS factors of the system.

Some of the avenues for further extensions are with larger and heterogeneous networks. Back tracking can be applied on attack flows to reach the attack source. Both of them hold promise for evaluating and improving our DDoS detection and defense method and data center information protection. The data center load can be still minimized by used some distributed load sharing algorithms.

References Références Referencias

1. wei yu, nan zhang, xinwen fu, Riccardo bettati, and wei zhao, "localization attacks to internet threat monitors:Modeling and countermeasures"on ieee transactions on computers, vol. 59, no. 12, december 2010.
2. J. Mirkovic and P. Reiher, "A Taxonomy of DDOS Attack and DDOS Defense Mechanisms," ACM SIGCOMM Computer Comm.Rev., vol. 34, no. 2, pp. 39-53, Apr. 2004.
3. SANS, Internet Storm Center, http://isc.sans.org/, 2010.
4. D. Moore, G.M. Voelker, and S. Savage, "Inferring Internet Deny-of-Service Activity," Proc. 10th USNIX Security Symp. (SEC), Aug. 2001.
5. Yegneswaran, P. Barford, and S. Jha, "Global Intrusion Detection in the Domino Overlay System," Proc. 11th IEEE Network and Distributed System Security Symp. (NDSS), Feb. 2004.
6. M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson, "The Internet Motion Sensor: A Distributed Blackhole Monitoring System," Proc. 12th Ann. Network and Distributed System Security Symp. (NDSS), Feb. 2005.
7. J. Bethencourt, J. Frankin, and M. Vernon, "Mapping Internet Sensors with Probe Response Attacks," Proc. 14th USNIX Security Symp. (SEC), July/Aug. 2005.
8. Y. Shinoda, K. Ikai, and M. Itoh, "Vulnerabilities of Passive Internet Threat Monitors," Proc. 14th USNIX Security Symp. (SEC), July/Aug. 2005.
9. X. Wang, W. Yu, X. Fu, D. Xuan, and W. Zhao, "Iloc: An Invisible Localization Attack to Internet Threat Monitoring Systems," Proc.IEEE INFOCOM (Mini-Conf.), Apr. 2008.
10. S. Cabuk, C. Brodley, and C. Shields, "Ip Covert Timing Channels:Design and Detection," Proc. 2004 ACM Conf. Computer and Comm.Security (CCS), Oct. 2004.
11. E. Cooke, F. Jahanian, and D. McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets," Proc. Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI), July 2005.
12. F.C. Freiling, T. Holz, and G. Wicherski, "Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks," Proc. 10th European Symp. Research in Computer Security (ESORICS), Sept. 2005.
13. The mstream distributed denial of service attack tool.http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.
14. J. Oikarinen and D. Reed. RFC 1459: Internet Relay Chat Protocol, 1993.
15. St´ephane Racine. Analysis of Internet Relay Chat Usage by DDoS Zombies. Master's thesis, Swiss Federal Institute of Technology Zurich, April 2004.
16. The Honeynet Project. Know your enemy: rackingbotnets.http://www.honeynet.org/papers/bots/,March 2005.
17. Ying Xuan, Incheol Shin, My T. Thai, Member, and Taieb Znati, Member, Detecting Application Denial-of-Service Attacks: A Group-Testing-Based Approach IEEE transactions on parallel and distributed systems, vol. 21, no. 8, august 2010.