Analysis on Some Defences against SYN-Flood Based Denial-of-Service Attacks

Sau Fan LEE
(ID: 3484135)
Computer Science Department,
University of Auckland
Email: slee283@ec.auckland.ac.nz

Abstract

A denial-of-service (DoS) attack is a type of resource-depleting attack on a system such that the system will no longer be able to take further requests from legitimate users. Usually, this type of attacks is network-based and accomplished by network-traffic flooding or by exploiting known vulnerabilities that is present in the server's system. A well-engineered DoS attack is very difficult to detect and is therefore a great threat to the Internet security today. Because of this, it is virtually impossible to find a solution that will solve all possible problems caused by DoS attacks. Different counter-measures must be applied in different situations (and locations) to reduce the number of DoS attacks and the damage done by them. In recent years, numerous proposals have been made in an attempt to solve various problem areas of DoS attacks. In this paper, We will focus only in one area of DoS attacks, known as TCP SYN-flooding. We will examine and compare the effectiveness and efficiency of 2 different counter-measures against this type of attacks, namely GENESIS and SYN Cookies.

1. Introduction

TCP/IP has now become an essential communications protocol in both local and wide area networks. However, the current implementation of the Internet Protocol, IPv4, has many flaws in its design, especially in terms of security. One of the most difficult to solve problems that companies face today is to defend against DoS (Denial-of-Service) attacks. Web-servers are especially vulnerable to DoS attacks, and a lot of efforts and research have been spent on finding ways to prevent DoS attacks from reaching the web servers or to allow web servers to continue serving legitimate users even in the presence of DoS attacks.

In 1ate 1996, D. J. Bernstein from University of Illinois and Eric Schenk from University of Toronto have devised a method of allowing a web-server to continue serving pages even after its connection resources are depleted by DoS attacks. This method is called *SYN cookies* and was later proven useful and was adopted into some UNIX and Linux builds.

In early 2002, another Internet security analyst by the name of Steve Gibson have devised a similar counter-measure, called *GENESIS*, that will allow web servers to distinguish between legitimate and fake requests to the server in order to filter out bad requests generated by DoS attacks. Although Gibson claimed that he had developed the method completely independently with no knowledge of the former counter-measure, his own counter-measure utilize the same concept but with slightly different implementation. GENESIS has not gone through extensive third party analysis and is therefore not widely known. However, as Gibson's influence in the general public is quite large, I decided to analyse the GENESIS system so that other people may get some feedback on the effectiveness and efficiency of this system when considering of using it. Since GENESIS is very similar to SYN Cookies in their architecture, direct comparisons can be made in many cases. Hence, criticisms and analysis done by experts on SYN Cookies can be re-iterated here.

In the following sections, we will first begin with some basic definitions in *Section 1* that are relevant to our discussions in later sections, followed by descriptions on how each of the 2 systems works in *Section 2*. In *Section 3* we will analyze and compare the effectiveness and efficiency of the algorithms used in the 2 systems, and finish with conclusions in *Section 4*.

1. What is DoS?

DoS is an acronym for *Denial of Service* attacks. It is a generic term that refers to an resource-depleting attack on a server on a network or on the Internet such that the server would no longer be able to serve legitimate users. The term DoS does not define what mechanisms are being used in such an attack, as there are many ways a DoS attack can be achieved. and mechanisms to achieve this, and DoS does not itself does not define what, as there can be many ways of it is usually done either by exploiting a known vulnerability of the server software in order to crash or cripple the server, or by flooding the server with excessive amount of traffic such that all

available resources on the server are consumed, rendering the server incapable of serving further requests from legitimate users.

Classic DoS attacks consist of only one machine attacking the victim. Usually, a single machine is not powerful enough to consume all the bandwidth that the server has, therefore a bandwidth-depleting DoS attack is normally not feasible. However, Other forms of resource-depleting attacks can be done, such as exploiting known vulnerabilities of the server software in order to crash or cripple the server, or by exhausting the memory resource of the server through *TCP SYN-flooding*, which we will be explained in more details later.

DDoS, on the other hand, uses multiple computers to conduct an attack against a single server. DDoS stands for *Distributed Denial of Service* attacks. It is a distributed attack in the sense that an attacker first compromise a large number of intermediate hosts and then use them to attack the victim simultaneously. This kind of attack requires the user to first infiltrate a large number of other computers before a DDoS attack can be conducted. It is very difficult to trace the original attacker of a DDoS attack as the attacker himself usually does not participate in the attack, but rather, he let the compromised hosts do all the dirty work for him. DDoS attacks are now very common and are primarily used for consuming up the *bandwidth* resource of the server.

TCP SYN flooding is a type of DoS attack that generates many bad *TCP SYN* packets. In a normal connection between the client and the server using TCP, the client first initiate the connection using a TCP SYN packet, then the server responded with a *SYN/ACK* packet, and then finally the client returns an *ACK* packet to establish the connection. This is commonly known as a *3-way handshake*. In a TCP SYN attack, the client (attacker) initiates a 3-way handshake but never finishes it. In other words, it only ever sends TCP SYN packets, with no final ACK packets. This will cause the server to reserve a memory slot for each unfinished connection. Once the server's memory is filled up with unfinished connections due to flooding of these packets, the server will stop accepting any more incoming requests until these memory slots are freed. Usually, these TCP SYN packets uses bogus IP addresses to prevent tracing of the attacker. Non-existent IP addresses will also ensure no replies are returned to the server.

Although TCP SYN flooding can be conducted using one computer, it is more often done using multiple computers (DDoS). Moreover, advanced attacks such as **DRDoS** (*Distributed Reflection DoS*) can be conducted using TCP SYN flooding. DRDoS is similar to SMURF attacks except that TCP SYN packets are used in place of ICMP Echo requests. In other words, in a DRDoS attack, the attacker sends a large number of TCP SYN packets to intermediate hosts using the victim's IP address as the source address. The intermediate hosts (called reflectors) will in turn reply to the victim with a massive number of SYN/ACK packets. If the victim does not reply (due to overloaded traffic) within a certain period, the reflectors will think the packets may be lost somewhere in transit, so they will resend the SYN/ACK packets, which will add more traffic to the victim. DRDoS is a relatively new form of attack and is more difficult to prevent since the SYN/ACK packets are actually legitimate responses from the reflectors. The reflectors themselves may not be able to tell that they are participating in a DRDoS attacks as the volume of incoming TCP SYN packets are usually not high enough to be considered as flooding. This is because the TCP SYN packets are evenly distributed across multiple reflectors, where each reflector only receives a small share of these packets.

In the following sections, we will take all these types of attacks into consideration when analyzing GENESIS and SYN Cookies. There are also many other types of DoS attacks besides the ones we had discussed here, but we do not include them here as they are not relevant to our analysis of the 2 systems. For more information on other common types of DoS attacks, please refer to [1] and [2].

2. The Inner Workings of SYN Cookies and GENESIS

The idea of SYN cookies is that the server tries to store authentication information in the server sequence number of the SYN/ACK packets. This idea is similar to cookies used in web sites where the server stores information of the current session in a cookie and return it to the client together with the web page, hence the name SYN cookies. When the server replies with a SYN/ACK packet, it uses a specially designed formula to calculate the server sequence number (used as an authentication cookie) and passed it into the SYN/ACK packet. The cookie value calculated by the formula is determined by the source address, the source port, the client address, the client port, the client sequence number, client MSS, a counter that

changes approximately every minute and a secret value that changes at every server boot. These information are merged together and encrypted using the MD5 one-way hash. When the client later replies with the ACK packet to complete the connection establishment, theserver will verify the server sequence number of the ACK packet to ensure the client is a legitimate user. Since the cookie is encrypted with a one-way hash, it is difficult to reverse-engineer the cookie directly. The cookie formula is also designed so that it will not give out a slightly smaller number than a recent value between the same hosts and ports. This is to prevent confusion with older packets from previous connections that may still be around.

Under normal situations when a web server's memory resource is not exhausted, the SYN cookies enabled server will act like any other normal web servers. That is, it will allocate memory slots for each incoming TCP SYN request. However, once the memory slots are used up, it will not allocate any more slots for the connections of the incoming SYN packets. However, it will still send out the SYN/ACK packets with the cookie as the server sequence number, but all the fancy and versatile features of TCP such as large window scaling, selective ACKs, RFC1323 etc. will be disabled. When a client returns with the ACK packet, the server will reproduce the cookie and matches it with the cookie in the ACK packet. If the 2 cookies match, then the authentication is successful. Note that the server also records the time of the last memory-slot outage. If the outage is more than a while ago, the ACK packet will still be rejected even if the cookie has the correct value. This expiry time is used to invalidate any out-of-date cookies may have a same value as another cookie at that moment. Also, due to the fact that the first 2 part of the handshake (SYN and SYN/ACK) are not remembered, it is not possible for the server to resend lost SYN/ACK packets to the client. For more detailed information on SYN Cookies, please refer to [3].

GENESIS stands for *Gibson's ENcryption-Enhanced Spoofing Immunity System*. It uses the same idea of SYN cookies in that it does not allocate any memory slots for incoming SYN packet connections and uses authentication 'cookies' as the server sequence number. However, there are a few notable differences. Firstly, this method is used regardless of whether memory slots are available or not. In other words, memory slots are not used *at all* on the server. Secondly, the encryption function used is RC5 instead of MD5. Also, the cookie does not store any information on the MSS,

and there is no counter or secret value involved. Since there are no memory slots, the time of the last memory outage is no longer relevant and therefore not used. For moer detailed information, please refer to [4].

3.3. Analysis and Comparisons of SYN Cookies and GENESIS

SYN Cookies successfully allows a server to remain online even after its memory resource is depleted. This achieves the primary goal of the SYN Cookies. However, there are sacrifices made, as large window scaling, selective ACKs, RFC 1323, and resending of lost packets are no longer possible, making the server inefficient when the memory resource is depleted. However, the fact that these problems are acknowledged is a good thing as people will be aware of the risks involved when they choose to use this system. But all in all, the system is good and has served its goal. The only gripe I have is with its poor documentation. It is stored in a raw unprocessed list of emails instead of an official well-formatted document.

GENESIS, on the other hand, does not remember the connections at all, making the problems in SYN Cookies after memory depletion becoming permanent in GENESIS. Because of this, from a certain point of view, the server actually becomes worse than not using SYN protection at all, since its service is less than satisfied. Also, the exclusion of MSS in GENESIS makes the web server *very* inefficient, as the traffic speed cannot be optimized this way. Furthermore, the use of RC5 is somewhat less safe as it is not a ‘one-way’ encryption. This enables an attacker to find out the key of the encryption and use it for malicious means. Although the possibility of finding the key before it expires is very low, it is still not as safe as a one-way hashes like MD5. Another problem with RC5 is that it is significantly slower than MD5 as its encryption algorithm involves a few rounds of encryption, whereas MD5 is equivalent to a single-round encryption. So, in summary, GENESIS is not a very good solution against DoS attacks.

If we refer back to the list of relevant DoS attacks in Section 2, both SYN Cookies and GENESIS give (partial) solutions to DoS and DDoS attacks based on TCP SYN flooding. Note my use of the word *partial* in the brackets. This is due to the trade-offs involve in both systems when under DoS attacks. Also note that GENESIS will still have the trade-offs even when *not* under attack. In terms of DRDoS attacks, both systems will still serve their fuctions when they are the *victims* of the attacks.

However, when it comes to being a reflector of an attack against other sites, both systems will happily participate in the attack. So the solutions provided for DRDoS attacks in both systems only solve part of the problem. In terms of the documentation of GENESIS, it is very well-written, easy to understand, and targets the average user who has not much knowledge in computer science.

4. Conclusions

SYN Cookies, while having some trade-offs, are still good in defending against TCP SYN floods. GENESIS, on the other hand, can also defend against TCP SYN floods, but its permanent bad side effects/trade-offs renders it impractical to be used in high volume sites. Both systems do not completely solve all the problems posed by TCP SYN flooding, although the major problems are all solved.

References

[1] Brendel, Juergen. "Distributed Denial of Service - The current state and counter measures" *Slides from NZISF meeting, Esphion Ltd. Auckland, New Zealand*, February 2002.

[2] Lin, Denny Ping-Herng, "Survey of Denial of Service Countermeasures" [Paper Online] November, 2000;
Available from *http://www.lasierra.edu/~dlin/classes/cpsc433/cpsc433.htm*
Accessed May 28.

[3] Bernstein, Dan J. "SYN cookies" [article online] no date;
Available from: *http://cr.yp.to/syncookies.html*
Accessed April 2003.

[4] Gibson, Steve. "G.E.N.E.S.I.S. (Gibson's ENcryption-Enhanced Spoofing Immunity System)" [paper online] March 11, 2002;
Available from: *http://grc.com/r&d/nomoredos.htm*
Accessed April 2003.