Denial of Service Attacks and Countermeasures Analysis

1. Introduction

In recent years, Denial of Service attacks (DoS) have been becoming attackers' favorites. This type of attack seems to be much more serious since attackers can take advantage of distributed network environment to perform the so-called Distributed Denial of Service (DDoS). By this type of attack, hacker can degrade network availability globally with surprising negligible computing resource. Moreover, many attack tools are open for free on Internet so even unskilled hackers are able to flood networks by DoS attacks, and so prevent many value-added network services from serving. So far, the network designers have mainly focused on network performance, they do not much concern about network security issues. This fact leads to many holes in network protocols, topologies as well as network softwares. In response to DoS attacks, since it is significantly expensive to build up new, secure network protocols and network softwares. Then, mostly, countermeasures are come from re-configuration, software patches, monitoring tools, firewall and so on. This paper is aimed to analyze each type of DoS attacks and make a comparison. We also note that, by the nature of cryptography, it provides very useful services such as authentication, identification to fight against DoS attacks. It is now considered as long-term and promising countermeasures. Thus, it is worth to concentrate on such kind of approach.

2. Background Knowledge

2.1. What is DoS Attack?

A "denial-of-service" attack is characterized by an explicit attempt by attackers to legitimate users of a service from using that service. Examples include

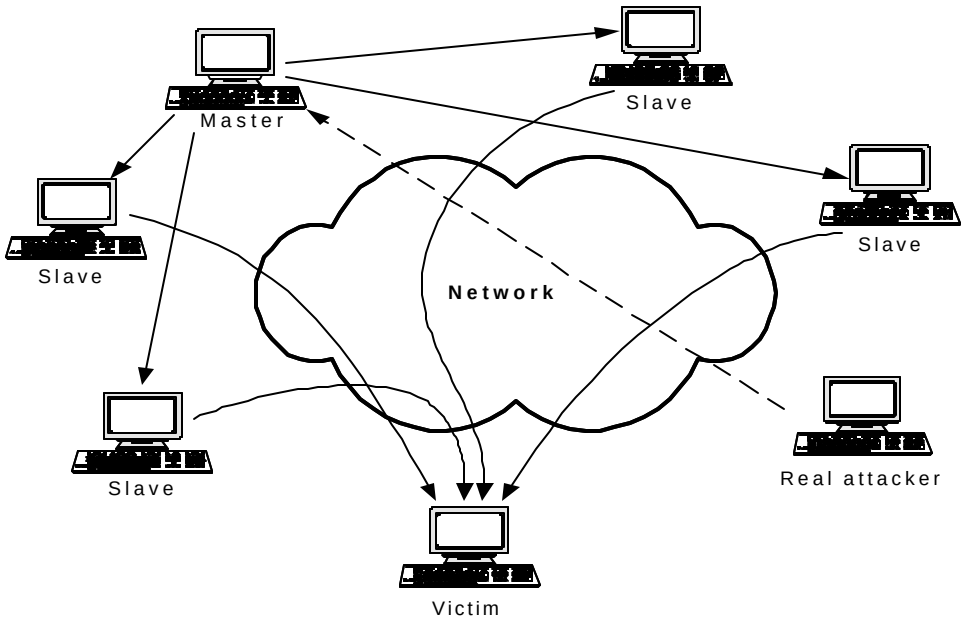
- attempts to *flood* a network, thereby preventing legitimate network traffic
- attempts to disrupt connections between two machines, thereby preventing access to a service
- attempts to prevent a particular individual from accessing a service
- attempts to disrupt to a specific system or person.

Not all service outages, even those that result from malicious activity, are necessarily denial-of-service attacks. Other types of attack may include a denial of service as a component, but the denial of service may be part of a larger attack.

Illegitimate use of resources may also result in denial of service. For example, an intruder may use your anonymous ftp area as a place to store illegal copies of commercial software, consuming disk space and generating network traffic.

Recently, there has come more serious form of DoS attack, known as distributed denial of service attack (DDoS attack). A distributed DoS attack amplifies the basic DoS attack. In a DDoS attack, attacker uses one computer to instruct many other computers to mount a powerful, coordinated attack (thus it is more difficult to fight against this type of attack). A typical DDoS attack consists of four components (Fig. 1): the real attacker, a control master, slaves (or attack daemon or zombies) and the victim. First, it involves a victim, i.e., the target host that has been chosen to receive the bunch of attacks. Second, it involves the presence of the attack daemon agents. These are agent programs that actually conduct the attack on the victim. Attack daemons are usually deployed in host computers. These daemons affect both the victim and the host computers. The task of deploying these attack daemons requires attacker to gain access and infiltrate the host computers. The third component of a distributed denial of service attack is the control master program. Its duty is to coordinate the attack. Finally, there is the real attacker, the hidden attacker behind the attack. By using one or many control master programs, the real attacker can stay behind the scenes of the attack. The following steps take place during a distributed DoS attack:

- The real attacker sends an "execute" message to the control master program.
- The control master program receives the "execute" order and then propagates the command to the attack daemons under its control.
- Upon receiving the attack command, the attack daemons begin the attack on the victim.



Although it seems that the real attacker has little to do but sends out the "execute" command, he actually has to plan the execution of a successful distributed denial of service attack. The attacker must infiltrate all the host computers and networks where the daemon attackers are to be deployed. The attacker must study the target's network topology and look for bottlenecks and vulnerabilities that can be exploited during the attack. Because the use of masters and slaves, the real attacker is not directly involved during the attack, which makes it difficult to trace who launched the attack.

2.2. DoS Impact

Denial of service attacks can essentially disable our computers or our networks. In early February, year 2000, hackers used distributed denial of service attacks that shut down some of the world's most high-profile websites, including Yahoo, Amazon.com, eBay, CNN.com, ZDNet, E*Trade and Excite.

Overall Internet Traffic slowed during three days of DoS attacks in Feb, 2000			
Date	Internet Performance (seconds)	Internet Performance a week earlier (seconds)	Change
7th February	5.98	5.66	5.7% slower
8th February	5.96	5.53	7.8% slower
9th February	6.67	5.26	26.8% slower
10th February	4.86	4.97	2.2% slower
			Source: Keynote Systems

Moreover, some denial-of-service attacks can be performed with limited resources against a complex site. This type of attack is sometimes called an "asymmetric attack".

3. DoS Attacks

3.1. Mode of attacks

Denial-of-service attacks come in a variety of forms and aim at a variety of services. There are three basic types of attack.

Consumption of Scarce, limited or non-renewable Resources

Computers and networks need certain things to operate: network bandwidth, memory and disk space, CPU time, data structures, access to other computers and networks, and certain environmental resources such as power, cool air, or even water.

1. Network Connectivity

Denial-of-service attacks are most frequently executed against network connectivity. The goal is to prevent hosts or networks from communicating on the network.

We should note that this type of attack does not depend on the attacker being able to consume your network bandwidth. In this case, the intruder consumes kernel data structures involved in establishing a network connection. The implication is that an intruder can execute this attack from a low speed (e.g. dial-up) connection against a machine on a very fast network.

2. Using Your Own Resources Against You

An intruder can also use your own resources against you in unexpected ways. The result is that the two services consume all available network bandwidth between them. Thus, the network connectivity for all machines on the same networks as either of the targeted machines may be affected.

3. Bandwidth Consumption

An intruder may also be able to consume all the available bandwidth on your network by generating a large number of packets directed to your network. Further, the intruder need not be operating from a single machine; he may be able to coordinate or co-opt several machines on different networks to achieve the same effect.

4. Consumption of Other Resources

In addition to network bandwidth, intruders may be able to consume other resources that your systems need in order to operate. For example, in many systems, a limited number of data structures are available to hold process information (process identifiers, process table entries, process slots, etc.). An intruder may be able to consume these data structures by writing a simple program or script that does nothing but repeatedly create copies of itself. Many modern operating systems have quota facilities to protect against this problem, but not all do. Further, even if the process table is not filled, the CPU may be consumed by a large number of processes and the associated time spent switching between processes. Consult your operating system vendor or operating system manuals for details on available quota facilities for your system.

An intruder may also attempt to consume disk space in other ways, including

- generating excessive numbers of mail messages.
- intentionally generating errors that must be logged
- placing files in anonymous ftp areas or network shares.

In general, anything that allows data to be written to disk can be used to execute a denial-of-service attack if there are no bounds on the amount of data that can be written.

Also, many sites have schemes in place to "lockout" an account after a certain number of failed login attempts. A typical set up locks out an account after 3 or 5 failed login attempts. An intruder may be able to use this scheme to prevent legitimate users from logging in. In some cases, even the privileged accounts, such as root or administrator, may be subject to this type of attack. Be sure you have a method to gain access to the systems under emergency circumstances. Consult your operating system vendor or your operating systems manual for details on lockout facilities and emergency entry procedures.

An intruder may be able to cause your systems to crash or become unstable by sending unexpected data over the network.

If your systems are experiencing frequent crashes with no apparent cause, it could be the result of this type of attack.

There are other things that may be vulnerable to denial of service attacks that administrators may wish to monitor. These include

- printers
- tape devices
- network connections
- other limited resources important to the operation of your organization

Destruction or Alteration of Configuration Information

An improperly configured computer may not perform well or may not operate at all. An intruder may be able to alter or destroy configuration information that prevents you from using your computer or network.

For example, if an intruder can change the routing information in your routers, your network may be disabled. If an intruder is able to modify the registry on a Windows NT machine, certain functions may be unavailable.

Physical Destruction or Alteration of Network Components

The primary concern with this type of attack is physical security. You should guard against unauthorized access to computers, routers, network wiring closets, network backbone segments, power and cooling stations, and any other critical components of your network.

Physical security is a prime component in guarding against many types of attacks in addition to denial of service. For information on securing the physical components of your network, we encourage you to consult local or national law enforcement agencies or private security companies.

3.2. Methods of Denial of Service Attacks

Because of pro-performance design of network and inexperienced IT workers, hackers can mount attacks by taking advantage of network software bugs and network protocol problems. Many types of networking software cannot cope with malformed Internet Protocol packets. When being hit by such packets, the networking software crashes.

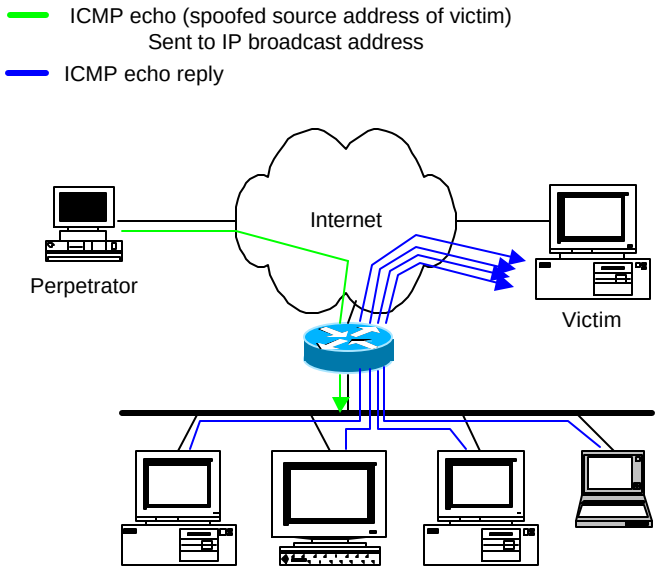
We will describe some well-known DoS attacks in following subsections.

A. Basic Denial of Service Attacks

Smurf. TCP/IP protocols provide facilities to help network managers or users identify network problems. One of the most frequently used debugging tools (*ping* program) invokes the ICMP (Internet Control Message Protocol) *echo request* and *echo reply* messages. A host or gateway sends an ICMP echo request message to a specified destination. Any machine that receives an echo request formulates an echo reply and returns it to the original sender.

Smurf attack is named after its program. In this attack, attackers can take advantage of this facility to attack a specific host by broadcasting ICMP echo request message to a large number of hosts. But the source address in the ICMP echo request is not attacker's machine's address; rather, it is victim's address (spoofed source address). Since every host should return ICMP echo reply to the original sender (specified by source address field in ICMP echo request message) whenever it receives the ICMP echo request message. Then, a large number of hosts accidentally return ICMP echo reply messages to the victim. If the attacker can produce ICMP echo request messages at very high rate (i.e. he owns a T.1 or T.3 connection), then, he can cause the victim to deny any legitimate connection due to high traffic load. Furthermore, on a multi-access broadcast network, there could potentially be hundreds of machines to reply to each ICMP echo request packet.

The illustration of this attack is given by the following picture (note that, the attacker is also called the perpetrator).

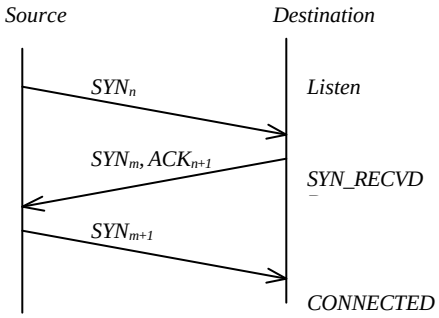


In practice, attackers usually steal a superuser account on a well-connected enterprise network to attack a powerful target. For a smaller target, he can use typical PPP dial-up account. Currently, the providers/machines most commonly hit are IRC servers (Internet Relay Chat servers) and their provider.

There are two parties who are hurt by this attack, the intermediary devices or *amplifiers* (broadcast), and the spoofed address target (the victim).

Finally, we can see that this type of attack is extremely feasible since it is very simple but powerful. Attackers only need to own a superuser account in a large enough co-location network to mount attacks to any host. Moreover, it is difficult to trace him because he uses the victim's address as source address in ICMP echo request message.

SYN Flood. As we know, TCP (Transmission Control Protocol) is a component of Internet protocol suite, which is on top of IP (Internet Protocol) layer. It provides a reliable, connection-oriented data stream delivery service. In TCP protocol, a connection is established by a procedure, called *three-way Handshake.*



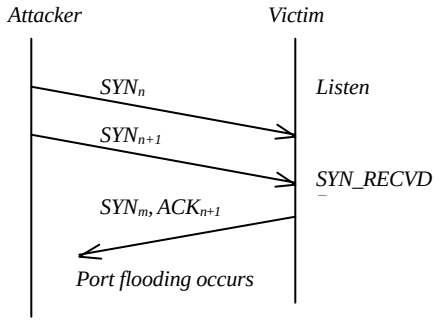
The first step in the procedure, the source host sends a SYN packet to the destination host. Then, the destination host sends a packet having both SYN and ACK flag set back to the source host. This means that the destination host acknowledges the SYN and is continuing the handshake. The final packet sent from the source host to the destination host has its ACK flag set indicating that both hosts agree that a connection has been established.

Note that, the three-way handshake also makes use of the sequence numbers for a new connection between source host and destination host. Sequence numbers are needed by TCP protocol to enable packet delivery and retransmission.

Clearly, any TCP connection needs some memory to store intermediate data. Under BSD style network code, there are three memory structures that need to be allocated by both endpoints, the *socket* structure, the *inpcb* structure and the *tcpcb* structure. When a SYN arrives at a port on which a TCP server is listening,

the three data structures allocated. There is a limit on the number of not-connected connections (or half-open connections or those connections which are in *SYN_RECVD* state). It is called backlog. When the maximum number of half-open connections is reached, TCP server will discard all new incoming connection requests until it either cleared or completed some of the half-open connections. It is also noted that there is expire time on a half-open connection. If the TCP server cannot complete or clear a half-open connection after a moment of time (usually 75 seconds), it will release memory allocated for this connection and turn back to *Listen* state.

In SYN flood attack, an attacker tries to send many connection requests with spoofed source address to victim's machine. That causes the victim's machine to allocate resources for those incoming connections and then sends ACK packets to the spoofed address. Since it sends ACK packets to a machine that does not involve in three-way handshake process then it will not receive final ACK packet to complete the connection (the spoofed address must not be reachable by the victim's machine in order to prevent the connection from being reset). Thus, all connections stay in half-open state. Once the limit of half-open connections is reached, the victim's machine will no longer accept new connections thus deny serving legitimate connections. Because the TCP server will clear half-open connections after a moment of time, then, in order to prolong the attack, the attacker will continuously send new connection requests to the victim's machine.



In a real attack, attackers usually construct a script with some parameters: the number of SYN packets per source address sent in a batch (batch-size), the break time between successive batches (delay) and the mode of source address allocation. The source address can be a single address or a short-list of addresses or randomly generated addresses.

We can see that in order to mount an attack, there must be a TCP port opened on the victim's machine and attackers are required to find a spoofed address that is not reachable by the victim's machine. Anyway, the amount of CPU and network bandwidth required by an attacker for a sustained attack is negligible. We also note that the basis of the attack is that TCP/IP does not provide strong authentication on its control packets. Furthermore, there is a requirement for an inappropriately burdensome allocation of memory and computation resources on the target side.

UDP Flood. UDP (User Datagram Protocol) is an alternative for TCP. It provides connectionless data stream delivery service. This type of attack is also known as *fraggle* and similar to Smurf attack. The only different is that an attacker uses *UDP echo* message (which expects *UDP reply* message) instead of ICMP echo message.

B. Distributed Denial of Service Attacks

Trinoo. As we described previously, in a distributed denial of service attack, an attacker tries to formulate groups of other hosts (master and daemon hosts) and control them to flood the victim. In the trinoo attack, the attacker manages to build up a so-called trinoo networks. After successfully installed masters and daemons and specified target (victim), the attacker can control the masters through TCP port 27665. Each master communicates with their daemons through UDP port 27444 (master to daemon) and 31335 (daemon to master). Each daemon can execute UDP flood against the victim. This type of attack aims at Sun Solaris and Linux operation system running various services known to have remotely exploitable buffer security bugs, such as wu-ftp (FTP Server), RPC services for "cmsd", "statd", "ttdbserverd", "amd",… Clearly, The more hosts are recruited by the attacker, the more powerful attack can be mounted by the him.

Tribe Flood Network (TFN) uses a command line interface to communicate between the attacker and the control master program. Communication between the control master and attack daemons is done via ICMP echo reply packets. TFN's attack daemons implement Smurf, SYN flood, UDP flood, ICMP flood attacks.

Stacheldraht (German term for "barbed wire") is based on the TFN attack. However, unlike TFN, Stacheldraht uses an encrypted TCP connection for communication between the attacker and master control program. Communication between the master control program and attack daemons is conducted using TCP and ICMP, and involves an automatic update technique for the attack daemons. The attack daemons for Stacheldraht implement Smurf, SYN Flood, UDP Flood, and ICMP Flood attacks.

Shaft is modeled after Trinoo. Communication between the control master program and attack daemons is achieved using UDP packets. The control master program and the attacker communicate via a simple TCP telnet connection. A distinctive feature of Shaft is the ability to switch control master servers and ports in real time, hence making detection by intrusion detection tools difficult.

TFN2K uses TCP, UDP, ICMP, or all three to communicate between the control master program and the attack daemons. Communication between the real attacker and control master is encrypted using a key-based CAST-256 algorithm. In addition, TFN2K conducts covert exercises to hide itself from intrusion detection systems. TFN2K attack daemons implement Smurf, SYN, UDP, and ICMP flood attacks.

4. Countermeasures

Firstly, we will introduce some basic solutions to cope with DoS attacks. Since, most of DoS attacks are based on bugs in network protocols and network software and attackers are required to steal a superuser account of a host computer in a network. Thus, a very natural way to prevent is to secure host computer from hacking. If we can guarantee high level of security for a host computer, we can prevent attacker from stealing superuser account and controlling it to attack the victim. Of course, such solution is very hard to guarantee.

We also see that, attacker always makes use of spoofed source address. So we can stop this by not allowing spoofed source address to reach that victim's machine. In order to do so, all networks should perform filtering either at the edge of network where customers connect (access layer) or at the edge of the network with connections to upstream providers, in order to defeat the possibility of source-address-spoofed packets from entering downstream networks, or leaving for upstream networks. Additionally, router vendors should provide feature to turn off the ability to spoof IP source address by checking the source address of a packet against the routing table to ensure the return path of the packet is through the interface it was received. In smurf attack, attacker tries to broadcast ICMP echo message to other hosts in the network. Thus, a router should have an option to disable receiving network-prefix-directed broadcasts on an interface and an option to disable forwarding network-prefix-directed broadcast. Another approach is to configure a router to limit certain types of traffic to specific sources and/or destinations.

Regarding TCP SYN flood attack, we can use several system configuration improvements to overcome TCP protocol security hole. Some of them are:

- Reduce timeout of half-open connections so that it will be cleared earlier. This will help in pruning half-open connections from TCP queue.
- Increase the backlog value to make system able to cope with more simultaneous half-open connections.
- Since, TCP SYN flood requires a TCP port opened on the victim's machine. Thus, we should disable non-essential services to reduce the number of ports can be attacked.

These countermeasures also have some shortcomings:

- Reducing timeout may deny legitimate access for machines to which the round trip times exceed the timeout period.
- Increasing backlog also results in increases in system resource usage.

Firewall, by its nature, is also another approach. Firewall is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from Internet-based attacks and to provide a single choke point

where security and audit can be imposed. For instance, in TCP SYN attack, Firewall can play as a RELAY. That is when a packet for a host behind the firewall is received, the firewall answers on its behalf. Only after the three-way handshake is successfully completed does the firewall contact the host and establish a second connection.

- In case of an attack, because the final ACK never arrives, the firewall terminates the connection and the host never receives the datagram. Of course, the firewall must not be vulnerable to TCP SYN flooding.
- In case of legitimate connection, after connection successfully established, the firewall creates a new connection to the host behind it on behalf of the original client. The firewall has to keep acting as a proxy to translate the sequence numbers in the packets that flow the client and the server (it can predict the sequence numbers to be used by the connection to quickly translate them).

Since all packets are required to be processed at firewall, so, this type of approach causes some packet delivery delay. The advantage of this approach is that the destination host never received spoofed source address packets.

Another way to use firewall is to use it as a semi-transparent gateway. The firewall lets SYN and ACK packets go through but monitor the traffic and react to it. When the internal host sends SYN+ACK, the firewall lets it through and generates and sends the ACK that moves the connection out of the backlog queue. If the firewall has not received the legitimate ACK after some arguably short period of time, it will send a RST (Connection reset) packet to the internal host, terminating the connection. If it receives the legitimate ACK, the firewall also lets it through, thus the destination host receives duplicate ACK packets. But TCP protocol is designed to cope with this situation, then, the duplicate ACK will be discarded and the data flow will continue without further firewall intervention. The main advantage of this approach over the previous one is that no delay is introduced for legitimate connections once they are established. The drawback of this method is that, in case of an attack, there are many open connections at the destination host. However, the limit on open connections (only depending on system resource available) is much larger than the limit of half-open connections. This approach also requires the timeout period to be carefully selected, so as not to deny access to legitimate hosts with long response time.

Now we will summarize several general countermeasures to DoS and DDoS attacks.

4.1 Detecting attacks

Even detecting that a service is under a denial of service attack can sometimes be difficult. Clients, which are denied of service naturally, detect it, but the condition isn't always easily noticeable at the server (e.g. TCP SYN flood attack).

Even when it has been determined that the service is indeed under attack, detecting which part of the incoming traffic belongs to the attack, and which is legitimate traffic, can be difficult. This problem is made more difficult by spoofed IP addresses.

Intrusion detection and reaction systems aim to cut off denial of service attacks by identifying the part of traffic which belongs to the attack, and denying service only to that part while continuing to serve legitimate clients. Most such mechanisms are very ad-hoc in nature; a determined attacker can fool them, and they can also produce false positives. For instance, a web proxy for a very large organization naturally produces a large number of traffic, which all looks like it's coming from a single or few IP addresses. To an intrusion detection system this might look like a flooding attack.

4.2 Tracing attacks

Cutting off an attack often requires tracing it to its source. The possibility of tracing probably also discourages attacks, since attackers know they are more likely to get caught. Thus, it can be thought as both preventive and reactive countermeasure.

Forging, or spoofing as it is usually called, of source IP address on the Internet is quite easy. Some techniques (such as cookies, described in the next section) can be used to get some level of assurance about the source IP address. In analysis of mechanisms for protecting confidentiality and integrity of messages, it is usually assumed that the attacker can modify, replay, and block any packets sent. This naturally allows trivial denial of service, so somewhat weaker assumptions are used when analyzing availability.

There have been several proposals for mitigating the problem of IP spoofing. Ingress filtering means filtering incoming IP addresses which should not occur on the correct link. For example, a central router at a university should filter out outgoing packets whose source address is not within the university's network.

This makes the network less likely to be used as a launching pad for attacks, and if deployed widely, should reduce the problem of IP spoofing.

Another proposal, ICMP traceback messages, attempts to trace back flooding attacks, even if they have forged source addresses. Other authors have proposed mechanisms for recording the route traveled in the packets themselves. Unfortunately, all of these require modifications to router software, so it remains to be seen if any of them will ever be widely deployed.

4.3 Cookies

Cookie is a piece of data (e.g. nonce), which is generated by the server and given to the client in the beginning of a protocol run. The client has to include this piece of data in the subsequent messages. The goal of cookies is to prevent attacks from employing IP spoofing. If the client doesn't receive the message containing the cookie, the server will reject further messages because they don't include a valid cookie. More precisely, if the server receives a valid cookie from the client, it knows that:

- The attacker is using his real IP address. This address may, of course, belong to a third party computer that the cracker has broken into.
- Or, the attacker has access to physical link on the route from the server to the spoofed IP. In reality, the attacker is probably quite close to either the server or the spoofed IP, making tracing easier.
- Or, the attacker is able to manipulate the IP routing infrastructure. This is beyond the capabilities of "script kids", and more sophisticated and motivated attackers probably focus on attacks on integrity and confidentiality.

  In other words, the idea is to start the protocol with weak authentication (of IP addresses), and possibly later perform stronger authentication. This allows tracing of attacks, and probably discourages attacks on computational resources.

An early example of cookies are actually the TCP initial sequence numbers, though their original purpose was to prevent packets from old connections interfering with new connections. After the TCP SYN attacks, some TCP/IP stacks were modified to use the initial sequence numbers as SYN cookies to protect against the attack. The cookie approach was much refined during the design of the Photuris protocol. The Photuris specification gives the following requirements for cookies (or anti-clogging tokens):

- The cookie must depend on the addresses of the communicating parties.
- Nobody else must be able to forge a cookie that will be accepted by the server.
- The cookie generation and verification must be fast enough so that they don’t become subjects to DoS attacks.
- The server must not keep per-client state until the IP address has been verified (i.e. it has received a cookie it generated).

The last requirement is especially important in protecting against memory consumption attacks. The recommended method for generating the cookies in Photuris is to use a keyed one way hash of both IP addresses, both UDP ports, some locally generated secret value (which must be same for all clients, and must be periodically changed), and some other context-dependent information.

4.4 Storing state in client

The cookie approach can be extended to include some states in the cookie. Any state that would be normally stored in the server is passed to the client. The client passes the state back to the server when sending the next message. The client doesn’t have to interpret the state in any way, and can treat it simply as an arbitrary bit string. Encryption and message authentication codes can be used to prevent the client from tampering with the state. This naturally doesn’t work for protocols where the server might be required to take some action before the reception of next message, but is otherwise a quite general approach. Stateless protocols have others advantages in addition to preventing memory consumption attacks. Stateless protocols also allows easier load balancing between servers.

4.5. Re-ordering computations

In typical authenticated versions of the Diffie-Hellman key agreement protocol, the server has to verify the client's signature in the first message. Since this requires expensive computation, the server can be potentially flooded with requests. In some cases, however, it is possible to modify the protocol so that client has to do some expensive computation first, and the server verifies the signature only after it has

verified that the client has done so. Thus, mounting an attack requires the client to invest the same amount of CPU resources as the server, and this hopefully will make DoS attacks at least somewhat harder.

4.6 Pricing

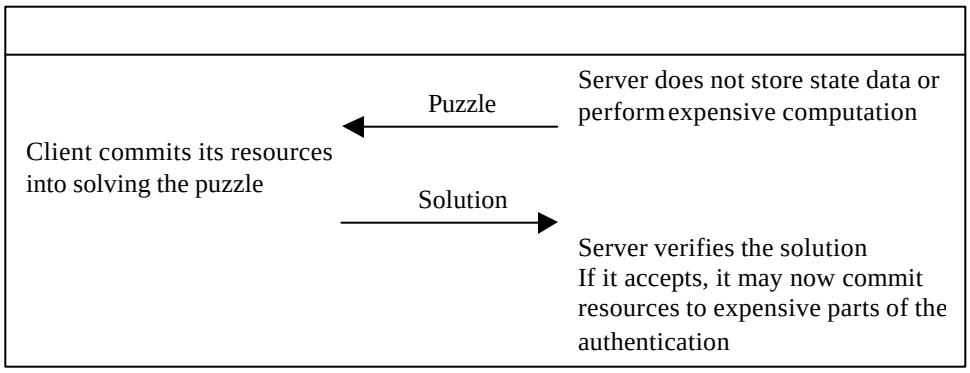
Although cookies, stateless connections and re-ordering computations can give some protection against DoS attacks, in some cases more aggressive measures are required to allow service to legitimate users and deny it to attackers. One such technique is "pricing". This means imposing some deliberate cost to the client, which is small for legitimate users making a small number of requests, but large for an attacker trying to flood the server. Earliest use of this approach was a proposal by Dwork and Naor to combat junk e-mail. Before accepting an e-mail message, the recipient asks the sender to perform a small computation (i.e. solving a small puzzle). The verification of the computation has to be quick, so this doesn't open possibilities for new DoS attacks. Another early use of this technique was against SYN flooding. Usually this cost is in terms of processing time, since it is easy to implement, but other forms (such as paying with actual micropayment systems) are also possible. The computation can be just "junk computation" to prevent denial of service, or it can be "useful" computation for some other purpose.

5. Protocols against DoS

As we can see, attackers mostly take advantage of lack of authentication in network protocol to mount DoS attack (i.e. address spoofing). Thus, we can think of constructing network protocol equipped with authentication feature to deny DoS attacks.

Recently, IPsec was proposed by Internet Engineering Task Force. IPsec ensures confidentiality, integrity and authenticity of data communication over a public IP network. It also provides an important component of a standard based, flexible solution for deployment of a network wide security policy. Since IPsec provides authentication service, we can expect that it will ignore DoS attacks too. But, there are many criticisms on complexity of IPsec. The design of IPsec obliviously tries to deal with many different situations with different operations. So, even though IPsec prevent attacker from using address spoofing, but it also introduces another kind of DoS attacks on its own. Attackers may exploit the complexity of protocol (i.e. expensive operation like exponentiation) to make host computer to abuse processor usage, thus, prevent host compute from serving legitimate users. Thus, authentication with less expensive computation or load balancing (i.e. client and server should be have same computation load) protocol is required.

Another approach is that the client should always commit its resources to the authentication protocol first and the server should be able to verify the client commitment before allocating its own resources. The rule is that at any point before reliable authentication, the cost of the protocol run to the client should be greater than to the server. The client's cost can be artificially increased by asking it to compute solutions to puzzles that are easy to generate and verify but whose difficulty for the solver can be adjusted to any level. The server should remain stateless (i.e. do not store and state) and refuse to perform expensive cryptographic operations until it has verified the client's solution to a puzzle. The strategy is described as follow.



A good puzzle should have the following properties,

- Creating a puzzle and verifying puzzle 's solution is inexpensive for the server.
- The cost of solving the puzzle is easy to adjust from zero to impossible (i.e. when server's resource is getting exhausted, server should increase the difficulty level).

- It is not possible to precompute solutions.
- While client is solving the puzzle, the server does not need to store the solution or other client specific data.
- The same puzzle may be given to several clients. Knowing the solution of one or more clients does not help a new client in solving the puzzle.
- A client can reuse a puzzle by creating several instances of it.

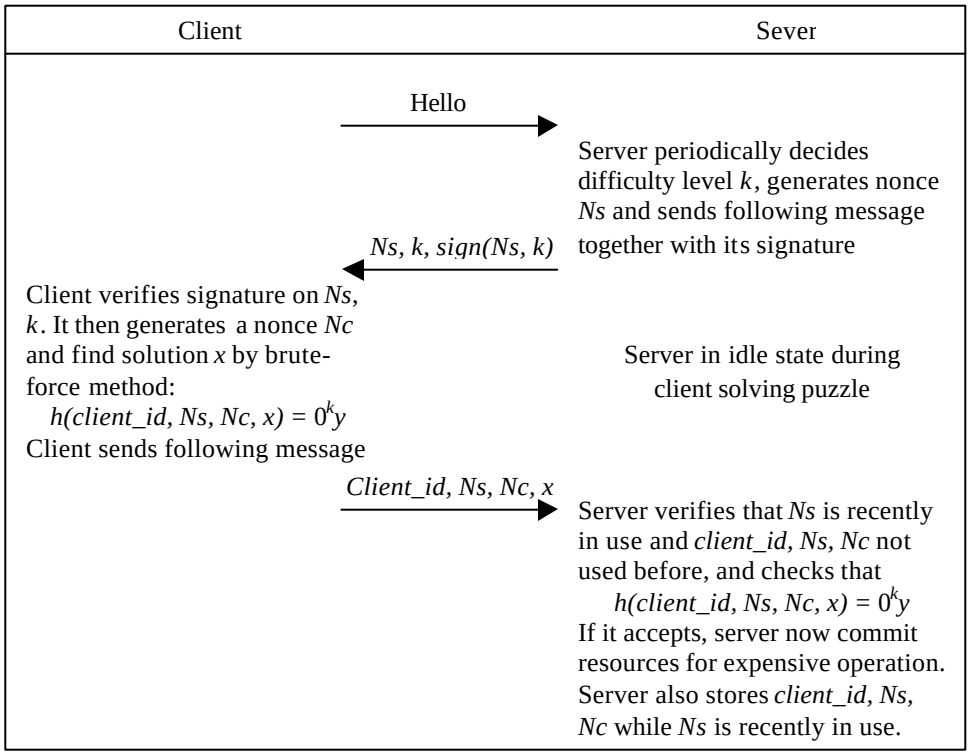
Next, we are going to show how to construct a puzzle satisfied above conditions by using one-way, collision-free hash function. The reason why we should use hash function is based on its security. Hash function is the simplest cryptographic primitive and it can be implemented in a wide range of hardware. Furthermore, hash function is free to use all over the world. A hash h function transfers the input of a arbitrary length bit string to a fixed length bit string (i.e. 160-bit bit string). The most important part of hash function is one-wayness and collision-free. The one-way property supports server in verifying the solution and forces client to do some expensive computation. The basic model of client puzzle is given by following equation:

$$H(Ns, x) = 0^k y$$

Where y is an arbitrary bit string of length $l - k$ (l is length of output of hash function), s is the puzzle server challenging the client, x is the solution to the puzzle that client must find out. The number k can be thought of difficulty level that is pre-determined by server. If k is large, then, it will be very difficult to find x given s because of one-way property of hash function. On the contrary, if k is small enough, then, it is feasible for the client to find x by brute-force attack. To prevent client form pre-computing the solution, s should be a nonce and it should be generated periodically. Note that, in order to prevent attacker from broadcasting false puzzles, server should sign on the puzzle he sends to clients. Also, each client should generate a nonce and its identity to put into input of hash function to make solution to a puzzle unique among other clients. Finally, we construct a puzzle as following equation:

$$h(client\_id, Nc, Ns, x) = 0^k y$$

Now, we will show the authentication protocol using client puzzle approach as follow:



6. Conclusion

In this paper, I gave an overview of all denial of service attacks and corresponding countermeasures. I described how each type of DoS attack works and how each type of countermeasure fight against DoS attacks. At last, I showed a cryptographic technique, so called client puzzle, to deny denial of service attack. Actually, we are always behind attackers in the way of fighting against DoS attacks. Thus, while pushing more efforts on constructing a practical complete solution to solve DoS attack problem, we should combine any known immediate solutions to protect cyber community from DoS attacks.

7. References

[1] http://www.cert.org

[2] Jussipekka Leiwo, *Towards Network Denial of Service Resistant Protocols.*

[3] Christoph L. Schuba, Ivan V.Krusl, Markus G. Kuhn, et al., *Analysis of a Denial of Service Attack on TCP.*

[4] Felix Lau, Stuart H. Rubin, Michael H. Smith, Ljiljana Trajkovic, *Distributed Denial of Service.*

[5] Tuomas Aura, Pekka Nikander, Jussipekka Leiwo, *DoS-Resistant Authentication with Client Puzzles.*

[6] Pasi Eronen, *Denial of Service In Public Key Protocols.*

[7] Douglas E. Comer, *Internetworking with TCP/IP, Principles, Protocols, and Architectures – Volume 1, Fourth Edition*

[8] RFC(s)

[9] David Dittrich et al, *The distributed denial of service attack tool series.*

[10] Niels Ferguson and Bruce Schneier, *A Cryptographic Evaluation of IPsec.*