Topology of Denial-of-Service

By Coretez Giovanni

Denial-of-Service (DOS) is defined by a service not being available to a person or process (application) when needed (availability). If the network processing is seen as a collection of information flows, then denial of service can be caused by three methods.

The first method is to take advantage of processes recovery (Designed Outage). Often devices, especially ones that have reset commands, temporarily shut themselves off or give control to another unrestricted process. For example, such an occurrence happens in the Windows environment when a network device is waiting. It is quite common to lock up an application to an improper DNS command waiting for an IP address that will never come, or an unconnected computer that has no chance of finding the DNS server. Most people have had the annoying sensation of looking at a web browser intent on reaching a page you know it cannot reach and the "stop" button is not living up to its name.

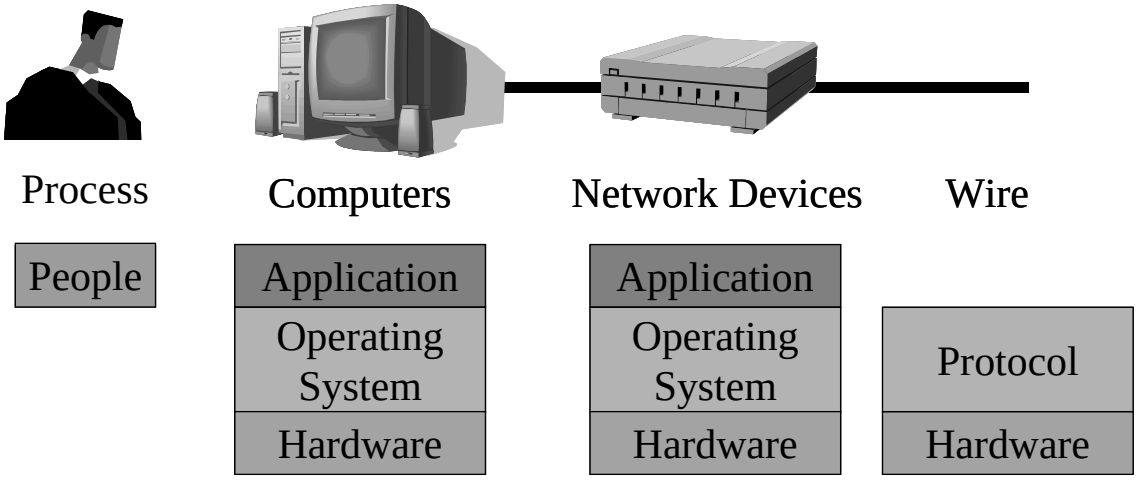
The second method is to "crash" a particular object in the information flow (Resource Destruction). By forcing an object in the process to become unstable and no longer functioning, the information it moves *can* be denied. The classic example people use of this type of denial is a backhoe cutting a telephone line. But this can be accomplished through buffer overflows that crash the service on the host.

And the last method is to overload a resource so that the information is slow and not there on time (Resource Exhaustion). A common example of this attack is a ping flood or a Trin00 attack.

Any element in the black and white figure below can be removed or overloaded to create a denial or service.

Designed Outage

Designed outages are just that, outages that are performed by the system to maintain the steady state. An example



of a designed outage occurs in Southern cities during the summertime. So many home cooling systems running during a hot day, that the power company is forced to turn off grids for a portion of time in order to maintain power to critical areas such as hospitals. Computer Networks and systems are constantly having designed outages. At the end of a TCP connection the system closes the quad (source IP, port, destination IP, port), and no longer accepts connections on that quad for a given length of time. Designed Outages occur to reduce error or device conflict. The last section in DOS addresses a designed outage for IP spoofing the rhost file.

Designed outages many not be desired and can be used maliciously. Consider the number of applications that "hang" the system while they are working. There are programs that require total control of a device or application regardless of if the process needs them. Malicious code can call such a processes to hang the system.

Resource Destruction

Destruction occurs when a program crashes. Operating systems are extremely complex, and not all mathematical possibilities can be predicted. Sometimes certain inputs trigger a system to crash. It is annoying for it happens all the time with graphical interface operating systems. It is possible to force a destructive state. This can occur remotely through an unusual packet (such as size or options) that which the socket code cannot handle. The socket hangs and in return, it hangs the system. Hanging the system is synonymous with the "Blue Screen of Death" for Window users.

In those good old days, graduate students were known to "IO Thrash" a VAX system the night before programs were due to have an excuse for not having projects done. IO Thrashing involved forcing the system to constantly load a new page in memory in an attempt to complete a process. These "Thrashing" programs were logical gridlocks that never allowed all the pages to be loaded and therefore the computer was unable to execute any of the programs in the queue.

Resource Exhaustion

Resource exhaustion occurs because computer systems are digital not analog. Digital systems work off a concept of mathematical ring theory.

```
unsigned int x
x = 0;
while (1) {
      printf("%d
\n", ++x);
}
```

The number x in this routine will hit a maximum integer value and start at 0 again. This is because all digital computation is based off mathematical rings. That means there is a finite number of anything in a computer. This causes problems when the system needs to number knowledge structures to keep track or access them.

Any process that requests more of a resource (memory, CPU, video, speakers) can be denied by simply hording all the possible numbers the system (or application) would use to address that resource. There are only so many files that a directory can have, and only so many files a computer can have. This is because the size of the look up table that points to the beginning of the file has a size limit:

```
while (1) do
x = time
Echo "$x" > time;
end do
```

So the proceeding pseudo-code would eventual take up all the directory space and crash the system. Some multi-users systems control the amount of space that a user is allowed access to protect against this (and to charge them for more space). During college classes that required threads (spawning other processes), the system administrators routinely brought down the system

every hour to kill the fatherless processes that had the system crawling.

Distributed attacks are a form of Resource Exhaustion. The most common is a bandwidth flood. Bandwidth flooding is a technique of using larger bandwidth to deny service to a small bandwidth owner. A wire like a hard drive can contain only so much data. A system that resides on a T-1 line can be flooded by a system of equal or greater size. A hacker needs to compromise a system with these characteristics to be successful. In distributed attacks, the bandwidth of three or four T-1 systems combine their bandwidth flooding to take out a T-3 accessed system.

It is important to understand that the result of the technique causes denial-of-service, but that the effect of denial-of-service can create some secondary effect that can cause compromise or some other collateral damage in the network.

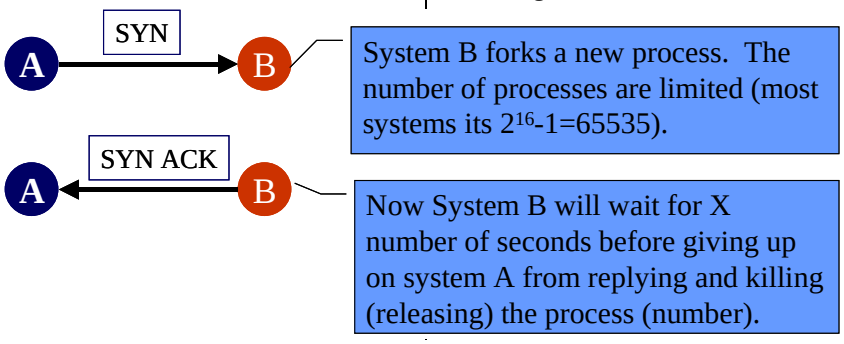
A later version of TFN2K could be used to accomplish knocking out a specifically defense asset, IDS analysts. TFN2K allowed for fake attack signatures to be placed into the flooding stream. Correctly done the attack could be used to create enough alarms in a network to make finding real alarms difficult.

Putting it Together

Now that there is a small-defined topology of DOS, let us attempt to determine a DOS situation for exploiting. Consider machine "A" sending a "SYN" packet to machine "B". We chose this because we know that there is a finite number of processes that a system can have. We also know that sending a SYN packet takes up a single process for most TCP oriented system services (the Listener forks a child process to enable listening to the socket and perform the application interface). This means by our definitions that it should be possible to create a situation of resource exhaustion.

Now if enough systems send "SYN" packets to machine B, then machine B will run out of processes to assign. And the system will no longer allow new processes until a new one is freed. So how quickly must the attacking system send data to the target system? Consider a resource release of 120 seconds. This means about 547 packets need to be sent a second. At 56 bytes a packet (448 bits) that comes to 245k/bits per second. So, this attack needs only ¼ of a T-1 line to launch and attack.

Conversely, this attack against a corporate network can freeze a particular system in the network without killing the entire bandwidth into the network. So, if a firewall allows access when it (the firewall) freezes or fails, then this attack can DOS the firewall and allow for an uninhibited attack against the network. If there is an IDS that can be detected, it too can temporarily be removed from the defense. This technique of eliminating a system but maintaining network connection is called *blocking*. In the next example, you will see blocking used again implemented as a designed outage DOS.

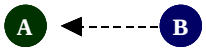


Denial of Service Compromise using Designed Outage

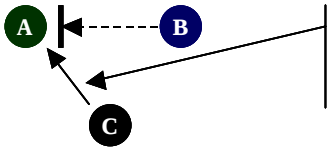
Denial-of-Service attacks are often considered a nuisance attack because they are not seen as a technique that will allow for the compromise of a system.

However, a denial-of-service attack can serve as the basis for a multi-phase attack such as IP spoofing. This is done through a sequence guessing technique. The most common form of the IP spoofing attack is to place a "++" into the ".host" file. This "++" symbol will allow entry for a root user from any site where the user is root. The attacker needs only to get the system to execute one command 'echo "++" >> .rhost'.

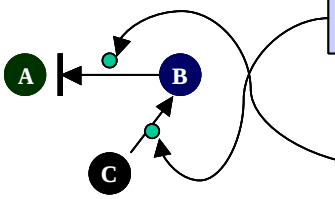
What this example shows is that denial of service can be used to gain access. This is another implantation of "Blocking". Blocking allows the removal of one element in the network from interacting with other elements. Blocking is useful in two cases, when the removal of the item allows entry in the system (as in above) or the removal of the element forces the system to use a different defensive technique that is not as secure as the primary defense.



System B trusts system A



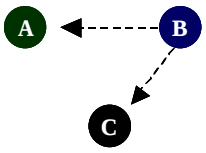
System C sends a series of NAK packets to A with an address from B. System A closes the port for a time to clear the channel by ignoring connection.



Later when C sends a fake packet to B, B will send an acknowledgement to A.

This acknowledgement will not receive a not acknowledged (NAK) because C has forced A to close that channel.

System C then sends a series of ACK packets to discover a pattern in the next sequence number that B uses. Once C determines a guess at the next sequence number. C sends a SYN packet, an ACK packet and a TCP PUSH packet with the command 'echo "++" >> .rhost'
System B's SYN ACK packet and later ACK packets never get a NAK from System A.



The executed command creates a new trust relationship that allows System C to rlogin into System B. This is accomplished by being able to guess the sequence number and being able to deny System A service to B.