Internet Engineering Task Force **IAB**
INTERNET-DRAFT **Mark Handley (ed)**
draft-iab-dos-00.ps **9 January 2004**
Expires: July 2004

Internet Denial of Service Considerations

Abstract

This document provides an overview of possible avenues for denial-of-service attack on Internet systems. The aim is to encourage protocol designers and network engineers towards designs that are more robust. We discuss partial solutions that reduce the effectiveness of attacks, and how some solutions might inadvertently open up alternative vulnerabilities.

1. Introduction

A Denial-of-Service (DoS) attack is an attack in which one or more machines target a victim and attempt to prevent the victim from doing useful work. The victim can be a network server, client or router, a network link or an entire network, an individual Internet user or a company doing business using the Internet, an Internet Service Provider (ISP), country, or any combination of or variant on these. Denial of service attacks may involve gaining unauthorized access to network or computing resources, but for the most part in this document we focus on the cases where the denial-of-service attack itself does not involve a compromise of the victim's computing facilities.

Because of the closed context of the original ARPAnet and NSFnet, no consideration was given to denial-of-service attacks in the original Internet Architecture. As a result, almost all Internet services are vulnerable to denial-of-service attacks of sufficient scale. In most cases, sufficient scale can be achieved by compromising enough end-hosts (typically using a virus, worm, or remotely controlled "bots") or routers, and using those compromised hosts to perpetrate the attack. Such an attack is known as a Distributed Denial of Service attack (DDoS). However, there are also many cases where a single well-connected end-system can perpetrate a successful DoS attack.

This document is intended to serve several purposes:

- To highlight possible avenues for attack, and by so doing encourage protocol designers and network engineers towards designs that are more robust.
- To discuss partial solutions that reduce the effectiveness of attacks.
- To highlight how some partial solutions can be taken advantage of by attackers to perpetrate alternative attacks.

This last point appears to be a recurrent theme in DoS, and highlights the lack of proper architectural solutions. It is our hope that this document will help initiate informed debate about future architectural solutions that might be feasible and cost-effective for deployment.

In addition it is our hope that this document will spur discussion leading to architectural solutions that reduce the succeptibility of all Internet systems to denial-of-service attacks.

We note that in principle it is not possible to distinguish between a sufficiently subtle DoS attack and a flash-crowd, where unexpected heavy but non-malicious traffic has the same effect as a DoS attack. Whilst this is true, such malicious attacks are usually more expensive to launch than many of the crude attacks that have been seen to date. Thus defending against DoS is not about preventing all possible attacks, but rather is largely a question of raising the bar sufficiently high for malicious traffic.

However, it is also important to note that not all DoS problems are malicious. Failed links, flash crowds, misconfigured bots, and numerous other causes can result in resource exhaustion problems, and so the overall goal should be to be robust to all forms of overload.

2. An Overview of Denial-of-Service Threats

In this section we will discuss a wide range of possible DoS attacks. This list cannot be exhaustive, but the intent is to provide a good overview of the spectrum of possibilities that need to be defended against.

We do not provide descriptions of any attacks that are not already publicly well documented.

2.1. DoS Attacks on End-systems

We first discuss attacks on end-systems. An end-system in this context is typically a PC or network server, but it can also include any communication endpoint. For example, a router also is an end-system from the point of view of terminating TCP connections for BGP [31] or ssh.

2.1.1. Exploiting Poor Software Quality

The simplest DoS attacks on end-systems exploit poor software quality on the end-systems themselves, and cause that software to simply crash. For example, buffer-overflow attacks might be used to compromise the end-system, but even if the buffer-overflow cannot be used to gain access, it will usually be possible to overwrite memory and cause the software to crash. Such vulnerabilities can in principle affect any software that uses data supplied from the network. Thus not only might a web server be potentially vulnerable, but it might also be possible to crash the back-end software (such as a database) to which a web server provides data.

Software crashes due to poor coding not only affect application software, but also the operating system kernel itself. A classic example is the so-called "Ping of Death", which became widely known in 1996 [10]. This exploit caused many popular operating systems to crash when sent a single fragmented ICMP echo request packet whose fragments totaled more than the 65535 bytes allowed in an IPv4 packet.

While DoS attacks such as the ping-of-death are a significant problem, they are not a significant architectural problem. Once such an attack is discovered, the relevant code can easily be patched, and the problem goes away. We should note though that as more and more software becomes embedded, it is important not to lose the possibility of upgrading the software in such systems.

2.1.2. Application Resource Exhaustion

Network applications exist in a context that has finite resources. In processing network traffic, such an application uses these resources to do its intended task. However, an attacker may be able to prevent the application from performing its intended task by causing the application to exhaust the finite supply of a specific resource.

The obvious resources that might be exhausted include:

- Available memory.
- The CPU cycles available.
- The disk space available to the application.
- The number of processes or threads or both that the application is permitted to use.
- The configured maximum number of simultaneous connections the application is permitted.

This list is clearly not exhaustive, but it illustrates a number of points.

Some resources are self-renewing: CPU cycles fall in this category - if the attack ceases, more CPU cycles become available.

Some resources such as disk space require an explicit action to free up - if the application cannot do this automatically then the effects of the attack may be persistent after the attack has ceased.

This problem has been understood for many years, and it is common practice for logs and incoming email to be stored in a separate disk partition (/var) on Unix systems.

Some resources are constrained by configuration: the maximum number of processes and the maximum number of simultaneous connections are not normally hard limits, but rather are configured limits. The purpose of such limits is clearly to allow the machine to perform other tasks in the event the application misbehaves. However, great care needs to be taken to choose such limits appropriately. For example, if a machine's sole task is to be an ftp server, then setting the maximum number of simultaneous connections to be significantly less than the machine can service makes the attackers job easier. But setting the limit too high may permit the attacker to cause the machine to crash (due to poor OS design in handling resource exhaustion) or permit livelock (see below), which are generally even less desirable failure modes.

2.1.3. Operating System Resource Exhaustion

Conceptually OS resource exhaustion and application resource exhaustion are very similar. However, in the case of application resource exhaustion, the operating system may be able to protect other tasks from being affected by the DoS attack. In the case of the operating system itself running out of resources, the problem may be more catastrophic.

Perhaps the best-known DoS attack on an operating system is the TCP SYN-flood [8], which is essentially a memory-exhaustion attack. The attacker sends a flood of TCP SYN packets to the victim, requesting connection setup, but then does not complete the connection setup. The victim instantiates state to handle the incoming connections. If the attacker can instantiate state faster than the victim times it out, then the victim will run out of memory that it can use to hold TCP state, and so it cannot service legitimate TCP connection setup attempts. This issue was exacerbated in some implementations by the use of a small dedicated storage space for half-open connections, which made the attack easier that it might otherwise have been. In the case of a poorly coded operating system, running out of resources may also cause a system crash.

An alternative TCP DoS attack is the Ack-flood [12], which is essentially a CPU exhaustion attack on the victim. The attacker floods the victim with TCP packets pretending to be from connections that have never been established. A busy server that has a large number of outstanding connections needs to check which connection the packet corresponds to. Some TCP implementations implemented this search rather inefficiently, and so the attacker could use all the victim's CPU resources servicing these packets rather than servicing legitimate requests.

We note that strong authentication mechanisms do not mitigate against such CPU exhaustion attacks. In fact poorly designed authentication mechanisms using cryptographic methods can exacerbate the problem. If such an authentication mechanism allows an attacker to present a packet to the victim that requires relatively expensive cryptographic authentication before the packet can be discarded, then this makes the attacker's CPU exhaustion attack easier.

CPU exhaustion attacks can be also be exacerbated by poor OS handling of incoming network traffic. In the absence of malicious traffic, an ideal OS should behave as follows:

- As incoming traffic increases, the useful work done by the OS should increase until some resource (such as the CPU) is saturated.
- From this point on, as incoming traffic continues to increase the useful work done should be constant.

However, this is often not the case. Many systems suffer from livelock [29] where, after saturation, increasing the load causes a decrease in the useful work done. One cause of this is that the system spends an increasing amount of time processing network interrupts for packets that will never be processed, and hence a decreasing amount of time is available for the application for which these packets were intended.

2.1.4. Attacks on Ongoing Communications

Instead of attacking the end-system itself, it is also possible for an attacker to disrupt ongoing communications. If an attacker can observe a TCP connection, then it is relatively easy for them to spoof packets to either reset that connection or to de-synchronize it so that no further progress can be made [25]. Such attacks are not prevented by transport or application-level security mechanisms such as TLS [20] or ssh, because the authentication takes place after TCP has finished processing the packets.

If an attacker cannot observe a TCP connection, but can infer that such a connection exists, it is theoretically possible to reset or desynchronize that connection by spoofing packets into the connection. However, in the absence of any other information, this would require an excessively large number of spoofed packets to guess both the port of the active end of the TCP connection (in most cases the port of the passive end is predictable) and the currently valid TCP sequence numbers. However, as some operating systems have poorly implemented predictable algorithms for selecting either the dynamically selected port or the TCP initial sequence number [38] [9], then such attacks may still be feasible.

An attacker might be able to significantly reduce the throughput of a connection by sending spoofed ICMP source quench packets, although most modern operating systems should ignore such packets. However, care should be taken in the design of future transport and signaling protocols to avoid the introduction of similar mechanisms that could be exploited.

2.1.5. Attacks using the Victim's Own Resources

Instead of directly overloading the victim, it may be possible to cause the victim or a machine on the same subnet as the victim to overload itself.

An example of such an attack is documented in [7], where the attacker spoofs the source address on a packet sent to the victim's UDP echo port. The source address is that of another machine that is running a UDP chargen server (a chargen server sends a character pattern back to the originating source). The result is that the two machines bounce packets back and forth as fast as they can, overloading either the network between them or one of the end-systems itself.

2.1.6. Triggered Lockouts and Quota Exhaustion

Many user-authentication mechanisms attempt to protect against password guessing attacks by locking the user out after a small number of failed authentications. If an attacker can guess or discover a user's ID, they may be able to trigger such a mechanism, locking out the legitimate user.

Another way to deny service using protection mechanisms is to cause a quota to be exhausted. This is perhaps most common in the case of small web servers being commercially hosted, where the server has a contract with the hosting company allowing a fixed amount of traffic per day. An attacker may be able to rapidly exhaust this quota, and cause service to be suspended. Similar attacks may be possible against other forms of quota.

In the absence of such quotas, if the victim is charged for their network traffic, a financial denial-of-service may be possible.

2.2. DoS Attacks on Routers

Many of the denial-of-service attacks that can be launched against end-systems can also be launched against the control processor of an IP router. In the case of a router, these attacks may cause the router to reboot, or may cause the router to cease processing routing packets. Even if the router does not stop servicing routing packets, it may become sufficiently slow that routing protocols time out. In any of these circumstances, the consequence of routing failure is not only that the router ceases to forward traffic, but also that it causes routing protocol churn that may have further side effects.

An example of such a side effect is caused by BGP route flap damping [33], which is intended to reduce global routing churn. If an attacker can cause BGP routing churn, route flap damping may then cause the flapping routes to be suppressed [27].

A DoS attack on the router control processor might also prevent the router being managed effectively. This may prevent actions being taken that would mitigate the DoS attack, and it might prevent diagnosis of the cause of the problem.

2.2.1. Attacks on Routers through Routing Protocols

In addition to their roles as end-systems, most routers run dynamic routing protocols. The routing protocols themselves can be used to stage a DoS attack on a router or a network of routers. To inject routing information into a routing protocol, an attacker needs access either to a router or to a end-host on a subnet where a routing protocol is running. In the latter case, if the routing protocol running on that subnet is not authenticated, the end-host may be able to insert spoofed routing information or pretend to be a router.

The simplest attack on a network of routers is to overload the routing table with sufficiently many routes that the router runs out of memory, or the router has insufficient CPU power to process the routes [15]. We note that depending on the distribution and capacities of various routers around the network, such an attack might not overwhelm routers near to the attacking router, but might cause problems to show up elsewhere in the network.

Some routing protocols allow limits to be configured on the maximum number of routes to be heard from a neighbor [17] limit doesn't block the spoofed routes at source, then imposition of such a limit elsewhere in the network might cause legitimate routes to be dropped.

An alternative attack is to overload the routers on the network by creating sufficient routing table churn that routers are unable to process the changes. Many routing protocols allow damping factors to be configured to avoid just such a problem. However, as with table size, such a threshold applied inconsistently may allow the spoofed routes to merge with legitimate routes before the mechanism is applied, causing legitimate routes to be damped.

The simplest routing attack on a specific destination is for an attacker to announce a spoofed desirable route to that destination. Such a route might be desirable because it has low metric, or because it is a more specific route than the legitimate route. In any event, if the route is believed it will cause traffic for the victim to be drawn towards the attacking router, where it will typically be discarded.

A more subtle denial-of-service attack might be launched against a network rather than against a destination. Under some circumstances, the propagation of inconsistent routing information can cause traffic to loop. If an attacker can cause this to happen on a busy path, the looping traffic might cause significant congestion, as well as not reaching the legitimate destination. However in many cases severe congestion is unlikely because TCP's congestion control will shut down the majority of traffic that fails to reach the destination.

If an attacker has access to a host on the same subnet as a router running certain routing protocols, even if that router is configured not to accept routes, it might be possible to cause that router to run out of memory by spoofing the existence of fake routers on that subnet.

In the past there have been cases where different generations of router interpreted a routing protocol specification differently. In particular, BGP specifies that in the case of an error, the BGP peering should be dropped. However, if some of the routers in a network treat a particular route as valid and other routes treat the route as invalid, then it may be possible to inject a BGP route at one point in the Internet and cause peerings to be dropped at many other places in the Internet. Unlike many of the examples above, while such an issue might be a serious short-term problem, this is not a fundamental architectural problem. Once the problem is understood, deploying patched routing code can permanently solve the issue.

2.2.2. IP Multicast-based DoS Attacks

There are essentially two forms of IP multicast: "traditional" IP multicast (ASM), as specified in RFC 1112 [19] where multiple sources can send to the same multicast group, and source-specific multicast (SSM) where the receiver must specify both the IP source address and the group address. The two forms of multicast provide rather different DoS possibilities.

With ASM, an attacker can simply send to multiple multicast groups. Routing protocols such as

PIM-SM [21], MSDP [28] and DVMRP [36] then have to instantiate routing state to ensure that the traffic goes to the group receivers and not to non-receivers. Thus ASM is particularly vulnerable to DoS attacks causing both multicast routing table explosion (and hence control processor memory exhaustion) and multicast forwarding table exhaustion (and hence forwarding card memory exhaustion or thrashing).

ASM also permits an attacker to send traffic to the same group as legitimate traffic, potentially causing network congestion and denying service to the legitimate group.

However, unlike unicast traffic, it is comparatively difficult to spoof source addresses for IP multicast traffic. Most deployed IP multicast routing protocols use reverse-path checks to build the forwarding tree, and so multicast attackers are easily located. In addition, multicast traffic will only continue to flow to a receiver as long as the receiver joins the multicast group. Thus it may be harder to use multicast traffic to cause a denial-of-service attack on a destination - an attacker would need to be opportunistic, sending traffic to a multicast group that is already being received by the victim or another host located close to the victim.

SSM does not permit senders to send to arbitrary groups unless a receiver has requested the traffic. Thus sender-based attacks on multicast routing state are not possible with SSM. However, as with ASM, a receiver can still join a large number of multicast groups causing routers to hold a large amount of multicast routing state, potentially causing memory exhaustion and hence denial-of-service to legitimate traffic.

2.2.3. Attacks on Router Forwarding Engines

Router vendors implement many different mechanisms for packet forwarding, but broadly speaking they fall into two categories: ones that use a forwarding cache, and ones that do not. With a forwarding cache, the forwarding engine does not hold the full routing table, but rather holds just the currently active subset of the forwarding table.

Routers or switches using a forwarding cache are potentially vulnerable if an attacker can send many packets to different destinations through the same router, causing the forwarding cache to thrash. One effect of this is likely to be that legitimate traffic is dropped because the cache entry has been lost and takes time to reinstate. Another possible effect is that the control traffic caused by the forwarding engine attempting to refresh the cache causes overload of the control processor (and potentially causes routing adjacencies to be dropped). In practice, this is only an issue if the forwarding engine does not have sufficient space for the full routing table. Even then such attacks may be difficult to perpetrate if the intended victim is not close to the attacker. In other cases such an effect would normally only be seen in the presence of a worm that manages to compromise a very large number of hosts, and then scans widely in its attempt to propagate further.

Many modern routers use a loosely coupled architecture, where one or more control processors handle the routing protocols, and communicate over an internal network link to special-purpose forwarding engines, which actually forward the data traffic. In such architectures it may be possible for an attacker to overwhelm the communications link between the control processor and the forwarding engine. This is possible because the forwarding engines support very high speed links, and the control processor simply cannot handle a similar rate of traffic.

There may be many ways in which an attacker can trigger communication between the forwarding engines and the control processor. The simplest way is for the attacker to simply send to the router's IP address, but this should in principle be relatively easy to prevent using filtering on the forwarding engines. Another way might be to cause the router to forward data packets using the "slow path". This involves sending packets that require special attention from the forwarding router; if the forwarding engine is not smart enough to perform such forwarding, then it will typically pass the packet to the control processor. In a router using a forwarding cache, it may be possible to overload the internal communications by thrashing the forwarding cache. Finally, any form of data-triggered communication between the forwarding engine and the control processor might cause such a problem. Certain multicast routing protocols including PIM-SM contain many such data triggered events that could potentially be problematic.

The effects of overloading such internal communications are hard to predict, and very implementation-dependent. One possible effect might be that the forwarding table in the forwarding engine gets out of synchronization with the routing table in the control processor that reflects what the routing protocols believe is happening. This might cause traffic to be dropped or to loop.

2.3. DoS Attacks on Local Hosts or Infrastructure

There are a number of attacks that might only be performed by a local attacker.

An attacker with access to a subnet may be able to prevent other local hosts from accessing the network at all by simply exhausting the address pool allocated by a DHCP server. This requires being able to spoof the MAC address of an ethernet or wireless card, but this is quite feasible with certain hardware and operating systems.

An alternative DHCP-based attack is simply to respond faster than the legitimate DHCP server, and to give out an address that is not useful to the victim.

These sort of bootstrapping attacks tend to be difficult to avoid because most of the time trust relationships are established after IP communication has already been established.

Similar attacks are possible through ARP spoofing [4]; an attacker can respond to ARP requests before the victim and prevent traffic from reaching the victim. Some brands of ethernet switch allow an even simpler attack - simply send from the victim's MAC address, and the switch will redirect traffic destined for the victim to the attacker's port.

It may be possible to cause broadcast storms [4] on a local LAN by sending a stream of unicast IP packets to the broadcast MAC address - some hosts on the LAN may then attempt to forward the packets to the correct MAC address greatly amplifying the traffic on the LAN.

802.11 wireless networks provide many opportunities to deny service to other users. Unless encryption is enabled, it is trivial to announce the presence of a basestation (or even ad-hoc mode host) with the same network name (SSID) as the legitimate basestation. Most host stacks don't deal gracefully with this.

Some 802.11 basestations have limited memory for the number of associations they can support. If this is exceeded, they may drop all associations.

Finally, as the authentication in 802.11 takes place at a comparatively high level in the stack, it is possible to simply deauthenticate or disassociate the victim from the basestation, even if WEP is in use [26]. Bellardo and Savage [3] describe some simple remedies that reduce the effectiveness of such attacks.

What all these attacks have in common is that they exploit vulnerabilities in the link auto-configuration mechanisms. Such problems are hard to solve because the reason for the existence of such autoconfiguration mechanisms is ease of use, and to secure them requires some form of authentication at a sufficiently early place in the autoconfiguration process.

2.4. DoS Attacks on Sites though DNS

In today's Internet, DNS is of sufficient importance that if access to a site's DNS servers is denied, the site is effectively unreachable, even if there is no actual communication problem with the site itself.

Many of the attacks on end-systems described above can be perpetrated on DNS servers. As servers go, DNS servers are not particularly vulnerable to DoS. So long as a DNS server has sufficient memory, a modern host can usually respond very rapidly to DNS requests for which it is authoritative. This was demonstrated in October 2002 when the root nameservers were subjected to a very large DoS attack [34]. A number of the root nameservers have since been replicated using anycast [1] to further improve their resistance to DoS. However it is important for authoritative servers to have relaying disabled, or it is possible for an attacker to force the DNS server to hold state [37].

Many of the routing attacks can also be used against DNS servers by targeting the routing for the server. If the DNS server is co-located with the site for which is authoritative, then the fact that the DNS server is also unavailable of secondary importance. However, if all the DNS servers are made unavailable, this may cause email to that site to bounce rather than being stored while the mail servers are unreachable, so distribution of DNS server locations is important.

Causing network congestion on links to and from a DNS server can have similar effects to end-system attacks or routing attacks, causing DNS to fail to obtain an answer, and effectively denying access to the site being served.

We note that if an attacker can deny external access to all the DNS for a site, this will not only cause email to that site to be dropped, but will also causes email *from* that site to be dropped. This is because recent versions of mail transfer agents such as sendmail will drop email if the mail originates from a domain that does not exist. This is a classic example of unexpected consequences. Sendmail performs this check as an anti-spam measure, and spam itself can be viewed as a form of DoS attack. Thus defending against one DoS attack opens up the vulnerability that allows another DoS attack.

Finally, a data corruption attack is possible if a site's nameserver is permitted to relay requests from untrusted third parties [37]. The attacker issues a query for the data he wishes to corrupt, and the victim's nameserver relays the request to the authoritative nameserver. The request contains a 16-bit ID that is used to match up the response with the request. If the attacker spoofs sufficient response packets from the authoritative nameserver just before the official response will arrive, each containing a forged response and a different DNS ID, then there is a reasonable chance that one of the forged responses will have the correct DNS ID. The incorrect data will then be believed and cached by the victim's nameserver, so giving the incorrect response to future queries. The probability of the attack can further be increased if the attacker issues many different requests for the same data with different DNS IDs, because many nameserver implementations will the issue relayed requests with different DNS IDs, and so the response only has to match any one of these request IDs [6] [32].

2.5. DoS Attacks on Links

The simplest DoS attack is to simply send enough non-congestion-controlled traffic that a link becomes excessively congested, and legitimate traffic suffers unacceptably high packet loss.

Under some circumstances the effect of such a link DoS can be much more extensive. We have already discussed the effects of denying access to a DNS server. Congesting a link might also cause a routing protocol to drop an adjacency if sufficient routing packets are lost, potentially greatly amplifying the effects of the attack. Good router implementations will prioritize the transmission of routing packets, but this is not a total panacea. If routers are peered across a shared medium such as ethernet, it may be possible to congest the medium sufficiently that routing packets are still lost.

Even if a link DoS does not cause routing packets to be lost, it may prevent remote access to a router using ssh or SNMP. This might make the router unmanageable, or prevent the attack being correctly diagnosed.

The prioritization of routing packets can itself cause a DoS problem. If the attacker can cause a large amount of routing flux, it may be possible for a router to send routing packets at a high enough rate that normal traffic is effectively excluded. This is however unlikely except on low bandwidth links.

Finally, it may be possible to an attacker to deny access to a link by causing the router to generate sufficient monitoring or report traffic that the link is filled. SNMP traps are one possible vector for such an attack, as they are not normally congestion controlled.

2.6. DoS attacks on firewalls

Firewalls are intended to defend the systems behind them against attack. In that they restrict the traffic that can reach those systems, then may also aid in defending against denial-of-service attacks. However, under some circumstances the firewall itself may also be used as a weapon in a DoS attack.

There are many different types of firewall, but generally speaking they fall into stateful and stateless classes. The state here refers to whether the firewall holds state for the active flows traversing the firewall. Stateless firewalls generally can only be attacked by attempting to exhaust the processing resources of the firewall. Stateful firewalls can be attacked by sending traffic that causes the firewall to hold excessive state or state which has pathological structure.

In the case of excessive state, the firewall simply runs out of memory, and can no longer instantiate the state required to pass legitimate flows. Most firewalls will then fail closed, causing denial-of-service to the systems behind the firewall.

In the case of pathological structure, the attacker sends traffic that causes the firewall's data structures to exhibit worst case behaviour. An example of this would be when the firewall uses hash tables to look up forwarding state, and the attacker can predict the hash function used. The attacker may then be able to cause a large amount of flow state to hash to the same bucket, which causes the firewall's lookup performance to change from O(1) to O(n), where n is the number of flows the attacker can instantiate [18]. Thus the attacker can cause forwarding performance to degrade to the point where service is effectively denied to the legitimate traffic traversing the firewall.

2.7. DoS attacks on IDS systems

Intrusion detection systems (IDS) suffer from similar problems to firewalls. It may be possible for an attacker to cause the IDS to exhaust its available processing power, to run out of memory, or to instantiate state with pathological structure. Unlike a firewall, an IDS will normally fail open, which will not deny service to the systems protected by the IDS. However it may mean that subsequent attacks that the IDS would have detected will be missed.

Some IDSs are reactive; that is on detection of a hostile event they react to block subsequent traffic from the hostile system, or to terminate an ongoing connection from that system. It may be possible for an attacker to spoof packets from a legitimate system, and hence cause the IDS to believe that system is hostile. The IDS will then cause traffic from the legitimate system to be blocks, hence denying service to it. The effect can be particularly bad if the legitimate system is a router, DNS server, or other system whose performance is essential for the operation of a large number of other systems.

2.8. DoS attacks on or via NTP

Network time servers are generally not considered security-critical services, but under some circumstances NTP servers might be used to perpetrate a DoS attack.

The most obvious such attack is to DoS the NTP servers themselves. Many end systems have rather poor clock accuracy and so, without access to network time, their clock will naturally drift. This can cause problems with distributed systems that rely on good clocks. For example one commonly used revision control system can fail if it perceives the modification timestamp to be in the future.

If the NTP servers relied on by a host can be subverted, either through compromising or impersonating them, then the attacker may be able to control the host's system clock. This can cause many unexpected consequences, including the premature expiry of dated resources such as encryption or authentication keys. This in turn can prevent access to other more critical services.

2.9. Physical DoS

The discussion thus far has centered on denial-of-service attacks perpetrated using the network. However, computer systems are only as resilient as the weakest link. It may be easier to deny service by causing a power failure, by cutting network cables, or by simply switching a system off, and so physical security is at least as important as network security.

2.10. Social Engineering DoS

The weakest link may also be human. In defending against DoS, the possibility of denial-of-service through social engineering should not be neglected, such as convincing an employee to make a configuration change that prevents normal operation.

2.11. Legal DoS

Computer systems cannot be considered in isolation from the social and legal systems in which they operate. This document focuses primarily on the technical issues, but we note that "cease and desist" letters, government censorship, and other legal mechanisms also touch on denial-of-service issues.

2.12. Spam and Black-hole Lists

Unsolicited commercial email, also known as "spam", can effectively cause denial-of-service to email systems. While the intent is not denial-of-service, the large amount of unwanted mail can waste the recipient's time, or cause legitimate email to fail to be noticed amongst all the background noise. If spam filtering software is used, some level of false positives is to be expected, and so these messages are effectively denied service.

One mechanism to reduce spam is the use of black-hole lists. The IP addresses of dial-up ISPs or mail servers used to originate or relay spam are added to black-hole lists. The recipients of mail choose to consult these lists and reject spam if it originates or is relayed by systems on the list.

One significant problem with such lists is that it may be possible for an attacker to cause a victim to be black-hole listed, even if the victim was not responsible for relaying spam. Thus the black-hole list itself can be a mechanism for effecting a DoS attack. Note that every black-hole list has its own policy regarding additions, and some are less susceptible to this DoS attack than others. Consumers of black-hole list technology are advised to investigate these policies before they subscribe.

3. Attack Amplifiers

Many of the attacks described above rely on sending sufficient traffic to overwhelm the victim. Such attacks are made much easier by the existence of "attack amplifiers", where an attacker can send traffic from the spoofed source address of the victim and cause a larger responses to be returned to the victim. A detailed discussion of such reflection attacks can be found in [30].

The simplest such attack was the "smurf" attack [11], where an ICMP echo request packet with the spoofed source address of the victim is sent to the subnet-broadcast address of a network to be used as an amplifier. Every system on that subnet then responds with an ICMP echo response that returns to the victim. Smurf attacks are no longer such a serious problem, as these days routers usually drop such packets and end-systems do not respond to them.

An alternative form of attack amplifier is typified by a DNS reflection attack. An attacker sends a DNS request to a DNS server requesting resolution of a domain name. Again the source address of the request is the spoofed address of the victim. The request is carefully chosen so that the size of the response is significantly greater than the size of the request, thereby providing the amplification. As an aside, it is interesting to note that the largest DNS responses tend to be those incorporating DNSsec authentication information. This attack amplifier can only be used by an attacker with the ability to spoof the source address of the victim. However, we note that if the victim's DNS server is configured to relay requests from external clients, it may be possible to cause it to congest its own incoming network link.

Another variant of attack amplifier involves amplification through retransmission. This is typified by TCP amplification attack known as "bang.c". The attacker sends a spoofed TCP SYN with the source address of the victim to a arbitrary TCP server. The server will respond with a SYN|ACK which is sent to the victim, and when no final ACK is received to complete the handshake, the SYN|ACK will be retransmitted a number of times. Typically this attack uses a very large list of arbitrarily chosen servers as reflectors. For the attack to be successful, the reflector must not receive a RST from the victim in response to the SYN|ACK - however if the attack traffic sufficiently overwhelms the server or access link to the server, then packet loss will ensure that many reflectors do not receive a RST in response to their SYN|ACK, and so continue to retransmit. The attack can be exacerbated by firewalls that silently drop the incoming SYN|ACK without sending a RST.

In general, the architectural lessons to be learnt are simple:

- As far as possible, perform ingress filtering [22] [35] to prevent source address filtering.
- Avoid designing protocols or mechanisms that can return significantly larger responses than the size of the request, unless a handshake is performed to validate the client's source address. Such a handshake needs to incorporate an unpredictable nonce that is secure enough to mitigate the amplification effects of the protocol.
- All retransmission during initial connection setup should be performed by the client.

4. DoS Solutions

Although it is not in principle possible to distinguish between a flash-crowd and a DoS attack, in practice it should be possible to raise the bar high enough that an attacker is forced to disguise their attack as legitimate traffic. This makes it much more expensive to perpetrate an attack. In addition, the goal should be to make it impossible to perpetrate an attack from an untraceable host. Thus, even though some attacks may be impossible to prevent, the victim would then have legal recourse. This does not however mean that anonymity should be impossible at the application level; merely that the choice to interact with an anonymous party should be at the discretion of the other party.

A guiding principle when hardening systems against DoS is to avoid causing additional different avenues for attack. A classic example is that of a mail server that verifies the "from" address of email is resolvable, which allows email from a site to be DoSed by merely preventing access to that site's DNS servers.

The guidelines below primarily concern the relatively obvious mechanisms that are already reasonably well understood. The aim of this document is to help protocol designers and network operators understand the state of the art, and to stimulate discussion on additional architectural solutions.

Don't Create an Attack Amplifier

The simplest guideline is to avoid building attack amplifiers and, as far as possible, to perform ingress filtering to prevent compromised systems on a subnet from spoofing the source address on packets.

Don't Hold State for Unverified Hosts

From an end-system server point of view, one simple aim is to avoid instantiating state without having completed a handshake with the client to validate their address, and as far as possible to push work and stateholding to client. There are a number of techniques that might be used to do this, including SYN-cookies [5] [2]. All client-server protocols should probably be designed to allow such techniques to be used, but the enabling of the mechanism should normally be at the server's discretion to avoid unnecessary work under normal circumstances.

State Lookup Complexity

Any system that instantiates per-connection state should take great care to implement the state-lookup mechanisms in such a way that performance can not be controlled by the attacker. One way to achieve this is to use hash-tables where the hash mechanism is keyed in such a way that the attacker cannot instantiate a large number of flows in the same hash bucket.

Avoid Livelock

Most operating systems use network interrupts to receive data from the network, which is a good solution if the host spends only a small amount of its time handling network traffic. However, this leaves the host open to livelock [29], where under heavy load the OS spends all its time handling interrupts and no time doing the work needed to handle the traffic at the application level. Server operating systems should consider using network polling at times of heavy load, rather that being interrupt-driven, and should be carefully architected so that as far as reasonably possible, traffic received by the OS is processed to completion, or very cheaply discarded.

Use Unpredictable Values for Session IDs

Most recent TCP implementations use fairly good random mechanisms for allocating the TCP initial sequence numbers. In general, any dynamically allocated value used purely to identify a communications session should be allocated using an unpredictable mechanism, as this increases the search space for an attacker that wishes to disrupt ongoing communications. Thus the dynamically allocated port of the active end of a TCP connection might also randomly allocated.

With DNS, the ID which is used to match responses with requests should also be randomly generated. However, as the ID field is only 16 bits, the protection is rather limited, especially in the face of birthday attacks.

Authenticate Routing Adjacencies

In general, cryptographic authentication mechanisms are too costly to form the main part in DoS prevention. However, routing adjacencies are too important to risk an attacker being able to inject bad routing information, which can affect more than the router in question. Additional non-cryptographic mechanisms should then be used to avoid arbitrary end-systems being able to cause the router to spend CPU cycles on validating authentication data.

For BGP, at the very least, this implies the use of TCP MD5 [24] or IPsec authentication, combined with the TTL 255 hack [23] to prevent EBGP association with non-immediate neighbours. In future, this will likely imply better authentication of the routing information itself.

Isolate Router-to-Router Traffic

As far as is feasible, router-to-router traffic should be isolated from data traffic. How this should be implemented depends on the precise technologies available, both in the router and at the link-layer. The goal should be that failure of the link for data traffic should also cause failure for the routing traffic, but that an attacker cannot directly send packets to the control processor of the routers.

A downside of this is that some diagnostic techniques (such as pinging consecutive routers to find the source of a delay) may no longer be possible. Ideally, alternative mechanisms (which do not open up additional avenues for DoS) should be designed to replace such lost techniques.

Graceful Routing Degradation

A goal with routing protocols is that of graceful degradation in overload, and automatic recovery after the source of the overload has been remedied. Some routing protocols satisfy this goal more than others. Although RIP doesn't scale well, if a router runs out of memory when receiving a RIP route, it can just drop the route and send an infinite metric to its peers. The route will later be refreshed, and if the original source of the problem has been resolved, the router will now be able to process it correctly.

On the other hand, BGP is stateful in the sense that a peer assumes you have processed or chosen to filter any route that it sent you. There is no mechanism to refresh state in the base BGP spec, and even the later route refresh option [16] is hard to use usefully in the presence of overload. A BGP router that cannot store a route it received has two choices: completely restart BGP, or shutdown one or more peerings [15]. This means that the effects of a BGP overload are rather more severe than they need to be, and so amplifies the effect of any attack.

In general, few routing protocol designs actively consider the possible behaviour of routers under overload conditions; this should be an explicit part of future routing protocol designs. Although precise details should clearly be left to implementors, the protocol design needs to give them the capability to do their job properly.

Source-Specific Multicast

Source-specific multicast is easier to manage than ASM, and opens up many fewer avenues for DoS attack. However, ASM has many uses for resource discovery and autoconfiguration. A prudent deployment strategy would be to use SSM for inter-domain multicast and ASM only within the more controlled environment of an intranet. In both environments, administrative controls should be set to prevent a single receiver from joining sufficient multicast groups to cause a state exhaustion problem in the multicast routers. Such a threshold needs to take into account that some multicast congestion control mechanisms use multiple multicast groups to allow the receiver to select an appropriate rate.

In the future, multicast protocols may be deployed which use shared bidirectional trees for interdomain multicast. Thus might allow ASM to be deployed without the DoS vulnerabilities exhibited by current inter-domain ASM solutions. Such solutions are not currently available.

Autoconfiguration and Authentication

Autoconfiguration mechanisms greatly ease deployment, and are increasingly necessary as the number of networked devices grows beyond what can be managed manually. However, it should be recognised that unauthenticated autoconfiguration opens up many avenues for attack. There is a clear tension between ease of configuration and security of configuration. Future autoconfiguration protocols should consider the need to allow different end-systems to operate at different points in this spectrum within the same autoconfiguration framework.

Establish a Monitoring Framework

Network operators are strongly encouraged to establish a monitoring framework to detect and log abnormal network activity. One can not defend against an attack one doesn’t detect or understand. Such monitoring tools can be used to set a baseline of "normal" traffic, and can be used to determine:

1. Aberrant flows.
2. Type and source of the aberrant flows.

This is extremely helpful when responding to DDoS or a flash crowd, and should be in place prior to the event.

5. Conclusions

In this document we have highlighted possible avenues for DoS attack on networks and networked systems, with the aim of encouraging protocol designers and network engineers towards designs that are more robust. We have discussed partial solutions that reduce the effectiveness of attacks, and highlighted how some partial solutions can be taken advantage of by attackers to perpetrate alternative attacks.

Our focus has primarily been on protocol and network architecture issues, but there are many things that network and service operators can do to lessen the threat. Further advice and information for network operators can be found in [13] [35] [14].

It is our hope that this document will spur discussion leading to architectural solutions that reduce the succeptibility of all Internet systems to denial-of-service attacks.

6. Acknowledgements

We are very grateful to Vern Paxson, Paul Vixie, Rob Thomas and Dug Song for their constructive comments on earlier versions of this document.

7. Authors' Addresses

Mark Handley
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom.

Email: M.Handley@cs.ucl.ac.uk

8. References

[1] J. Abley, "Hierarchical Anycast for Global Service Distribution", http://www.isc.org/tn/isc-tn-2003-1.txt

[2] T. Aura, P. Nikander, J. Leiwo, "DOS-resistant authentication with client puzzles", In B. Christianson, B. Crispo, and M. Roe, editors, Proceedings of the 8th International Workshop on Security Protocols, Lecture Notes in Computer Science, Cambridge, UK, April 2000.

[3] J. Bellardo, S. Savage, "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions", Proceedings of the USENIX Security Symposium, Washington D.C., August 2003.

[4] S.M. Bellovin, "Security Problems in the TCP/IP Protocol Suite", Computer Communication Review, Vol. 19, No. 2, pp. 32-48, April 1989.

[5] D.J. Bernstein, "SYN Cookies", http://cr.yp.to/syncookies.html

[6] CCAIS/RNP Alertas do Cais ALR-19112002a, "Vulnerability in the sending requests control of Bind versions 4 and 8 allows DNS spoofing", http://www.rnp.br/cais/alertas/2002/cais-ALR-19112002a.html

[7] CERT Advisory CA-1996-01, "UDP Port Denial-of-Service Attack", Feb 1996.

[8] CERT Advisory CA-1996-21, "TCP SYN Flooding and IP Spoofing Attacks", Sept 1996.

[9] CERT Advisory CA-2001-09, "Statistical Weaknesses in TCP/IP Initial Sequence Numbers", May 2001.

[10] CERT Advisory CA-1996-26, "Denial-of-Service Attack via ping", Dec 1996.

[11] CERT Advisory CA-1998-01, "Smurf IP Denial-of-Service Attacks", http://www.cert.org/advisories/CA-1998-01.html, Jan 1998.

[12] CERT Incident Note IN-2000-05, "'mstream' Distributed Denial of Service Tool", May 2000.

[13] CERT/CC - "Managing the Threat of Denial of Service Attacks", http://www.cert.org/archive/pdf/Managing_DoS.pdf

[14] CERT/CC - "Trends in Denial of Service Attack Technology", http://www.cert.org/archive/pdf/DoS_trends.pdf

[15] D.-F. Chang, R. Govindan, J. Heidemann, "An Empirical Study of Router Response to Large Routing Table Load", Proceedings of the 2nd Internet Measurement Workshop (IMW 2002), 2002.

[16] E. Chen, "Route Refresh Capability for BGP-4", RFC 2918, September 2000

[17] Cisco Systems, "Configuring the BGP Maximum-Prefix Feature", Cisco Document ID: 25160, http://www.cisco.com/warp/public/459/bgp-maximum-prefix.html

[18] Scott A Crosby and Dan S Wallach, "Denial of Service via Algorithmic Complexity Attacks", Proceedings of the USENIX Security Symposium, Washington D.C., August 2003.

[19] S. Deering, "Host extensions for IP multicasting", RFC 1112, Aug 1989.

[20] T. Dierks, C. Allen, "The TLS Protocol, Version 1.0", RFC 2246, Jan 1999.

[21] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2362, June 1998.

[22] P. Ferguson, D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC 2827, May 2000.

[23] V. Gill, J. Heasley, D. Meyer, "The BGP TTL Security Hack (BTSH)", draft-gill-btsh-02.txt (work in progress), 29-May-03.

[24] A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, August 1998.

[25] Laurent Joncheray, "Simple Active Attack Against TCP", 5th USENIX Security Symposium, 1995.

[26] M. Lough, "A Taxonomy of Computer Attacks with Applications to Wireless", PhD thesis, Virginia Polytechnic Institute, April 2001.

[27] Z. Mao, R. Govindan, G. Varghese, R. Katz, "Route Flap Dampening Exacerbates Internet Routing Convergence", Proceedings of ACM SIGCOMM, 2002.

[28] D. Meyer, W. Fenner (Editors), "Multicast Source Discovery Protocol (MSDP)", draft-ietf-msdp-spec-15.txt, Work in Progress.

[29] J. Mogul, KK. Ramakrishnan, "Eliminating Receive Livelock in an Interrupt-driven Kernel", ACM Transactions on Computer Systems, Vol 15, Number 3, pp. 217-252, 1997.

[30] V. Paxson, "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks", Computer Communication Review 31(3), July 2001.

[31] Y. Rekhter, T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995.

[32] Joe Stewart, "DNS Cache Poisoning - The Next Generation", Jan 27 2003, http://www.securityfocus.com/guest/17905

[33] C. Villamizar, R. Chandra, R. Govindan, "BGP Route Flap Damping", RFC 2439, November 1998.

[34] P. Vixie, G. Sneeringer, M. Schleifer, "Events of 21-Oct-2002", http://f.root-servers.org/october21.txt

[35] P. Vixie, "Securing the Edge", http://www.icann.org/committees/security/sac004.txt

[36] D. Waitzman, C. Partridge, S.E. Deering, "Distance Vector Multicast Routing Protocol", RFC 1075, Nov 1988.

[37] D. Wessels, "Running An Authoritative-Only BIND Nameserver", http://www.isc.org/tn/isc-tn-2002-2.txt

[38] M. Zalewski, "Strange Attractors and TCP/IP Sequence Number Analysis", http://razor.bindview.com/publish/papers/tcpseq.html