Network Working Group
Request for Comments: 4732
Category: Informational

M. Handley, Ed.
UCL
E. Rescorla, Ed.
Network Resonance
Internet Architecture Board
IAB

November 2006

Internet Denial-of-Service Considerations

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The IETF Trust (2006).

Abstract

This document provides an overview of possible avenues for denial-of-service (DoS) 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.

Table of Contents

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 or worm) or routers, and using those compromised hosts to perpetrate the attack. Such an attack is known as a Distributed Denial-of-Service (DDoS) attack. 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 susceptibility 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 [10] or ssh [46].

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 affect not only 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 [21]. 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) in order to limit the impact of exhaustion.

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 attacker's 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 [19], 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 than 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 [23], 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 necessarily 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 [33] 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. 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, for example, by flooding the command and control access ports. In the case of a router, these attacks may cause the router to stall, 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 [11], 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 [31]. This suppression likely causes the networks served by those routes to become unreachable.

A DoS attack on the router control processor might also prevent the router from 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. This requires the ability to send traffic from addresses that might plausibly have generated the relevant routing messages, which is somewhat difficult with interior routing protocols but fairly easy with External Border Gateway Protocol (eBGP), for instance.

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 [26]. 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 protocol implementations allow limits to be configured on the maximum number of routes to be heard from a neighbor [27].

However, limits often make the problem worse rather than better, by making it possible for the attacker to push out legitimate routes with spoofed routes, thus creating an easy form of DoS attack.

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 fail to reach the legitimate destination.

In the past, there have been cases where different generations of routers 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 routers 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 Any-Source Multicast (ASM), as specified in RFC 1112 [4] 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.

ASM protocols such as PIM-SM [6], MSDP [32], and DVMRP [12] typically cause some routers to instantiate routing state at the time a packet is sent to a multicast group. They do this to ensure that the traffic goes to the group receivers and not to non-receivers. Such protocols are particularly vulnerable to DoS attacks, as an attacker that sends to many multicast groups may cause 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.

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.

With IPv6, hosts are required to send ICMP Packet Too Big or Parameter Problem messages under certain circumstances, even if the destination address is a multicast address. If the attacker can place himself in the appropriate position in the multicast tree, a packet with an unknown but mandatory Destination Option, for instance, could generate a very large number of responses to the claimed sender.

With IPv4, the same problem exists with multicast ICMP Echo Request packets, but these are somewhat easier to filter.

The examples above should not be taken as exhaustive. These are actually specific cases of a general problem that can happen when a multicast/broadcast request solicits a reply from a large number of nodes.

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.

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 are 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.

Finally, if an attacker can generate traffic that causes a router to auto-install access control list (ACL) entries, perhaps by triggering a response from an intrusion detection system, then it may be possible to exhaust the ACL resources on the router. This might prevent future attacks from being filtered, or worse, cause ACL processing to be handled by the route processor.

2.3. 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 [29]. Such attacks are not

prevented by transport or application-level security mechanisms such as TLS [5] 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 de-synchronize that connection by spoofing packets into the connection. However, this might 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 [41] [20], then such attacks have been found to be feasible [34]. Advice as to how to reduce the vulnerability in the specific case of TCP is available in [37].

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.

2.4. 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 [18], 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.

2.5. 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 Dynamic Host Configuration Protocol (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 sorts 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 [16]; 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. This attack might also potentially be used to block traffic from the victim by engaging screening or flap-dampening algorithms in the switch, depending on the switch design.

It may be possible to cause broadcast storms [16] 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. In some cases, the lack of defenses against DoS was a deliberate choice--because 802.11 operates on unlicensed spectrum it was assumed that there would be sources of interference and that producing intentional radio-level jamming would be trivial. Thus, the amount of DoS protection possible at higher levels was minimal.

Nevertheless, some of the weaknesses of the protocols against more sophisticated attacks are worth noting. The most prominent of these is that association is unprotected, thus allowing rogue access points (APs) to solicit notifications that would otherwise have gone to legitimate APs.

The SSID field provides effectively no defense against this kind of attack. Unless encryption is enabled, it is trivial to announce the presence of a base station (or even of an ad-hoc mode host) with the same network name (SSID) as the legitimate basestation. Even adding authentication and encryption a la 802.1X and 802.11i may not help much in this respect. The SSID space is unmanaged, so everyone is free to put anything they want in the SSID field. Most host stacks don't deal gracefully with this. Moreover, SSIDs are very often set to the manufacturer's default, making them highly predictable.

Some 802.11 basestations have limited memory for the number of associations they can support. If this is exceeded, they may drop all associations. In an attempt to forestall this problem, some APs advertise their load so as to enable stations to choose APs that are less loaded. However, crude implementations of these algorithms can result in instability.

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 Wired Equivalent Privacy (WEP) is in use [30]. Bellardo and Savage [15] describe some simple remedies that reduce the effectiveness of such attacks. While IEEE 802.11w will protect Deauthenticate or Disassociate frames, this attack is still possible via forging of Association frames.

What all these attacks have in common is that they exploit vulnerabilities in the link auto-configuration mechanisms. In a wireless network, it is necessary for a station to detect the presence of APs in order to choose which one to connect to. In 802.11, this is handled via the Beacon and Probe Request/Response mechanisms.

Beacons cannot easily be encrypted, because the station needs to utilize them prior to authentication in order to discover which APs it may wish to communicate with. Since authentication can only occur after interpreting the Beacon, an encrypted Beacon would present a chicken-egg problem: you can’t obtain a key to decrypt the Beacon until completing authentication, and you may not be able to figure out which AP to authenticate with prior to decrypting the Beacon. Note that in principle you could encrypt Beacons with a shared (per-AP) key but this would require each station to trial-decrypt beacons until it finds one that matches up to whatever shared authentication secret it had. This is not particularly convenient.

As a result, discussions of Beacon frame security have largely focused on authentication of Beacon frames, not encryption. Even here, solutions are difficult. While it may be possible for a station to validate a Beacon *after* authentication (either by checking a Message Integrity Check (MIC) computed with the group key provided by the AP or verifying the Beacon parameters during the 4-way handshake), doing so *before* authentication may require synchronization of keys between APs within an SSID.

2.6. DoS Attacks on Sites through 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 [38]. 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 servers to hold state [40].

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 is 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 servers for a site, this will not only cause email to that site to be dropped, but it will also cause 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. If a receiving implementation is using a black-hole list (see Section 2.14) served by DNS, an attacker can also mount a DoS attack by attacking the black-hole server.

Finally, a data corruption attack is possible if a site's nameserver is permitted to relay requests from untrusted third parties [40]. 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 arrives, 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 issue relayed requests with different DNS IDs, and so the response only has to match any one of these request IDs [17] [36].

The use of anycast for DNS services makes it even more vulnerable to spoofing attacks. An attacker who can convince the ISP to accept an anycast route to his fake DNS server can arrange to receive requests and generate fake responses. Anycast DNS also makes DoS attacks on DNS easier. The idea is to disable one of the DNS servers while maintaining the BGP route to that server. This creates failures for any client that is routed to the (now defunct) server.

2.7. DoS Attacks on Links

The simplest DoS attack is to simply send enough non-congestion-controlled traffic such 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 Simple Network Management Protocol (SNMP) [48]. This might make the router unmanageable, or prevent the attack from 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. However, this is unlikely except on low-bandwidth links.

Finally, it may be possible for 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.

Attackers with physical access to multiple access links can easily bring down the link. This is particularly easy to mount and difficult to counter with wireless networks.

2.8. 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, they 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 that 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 disconnected, 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 [28]. Thus, the attacker can cause forwarding performance to degrade to the point where service is effectively denied to the legitimate traffic traversing the firewall.

2.9. DoS Attacks on IDS Systems

Intrusion detection systems (IDSs) 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 blocked, 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.

2.10. 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.

2.11. 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. Physical attacks can also serve as

entry points for non-physical DoS, for instance, by reducing the resources available to deal with overcapacity.

2.12. 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.

2.13. 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.

2.14. 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. Similar considerations apply to feeds of bad BGP bad route advertisements.

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 larger responses to be returned to the victim. A detailed discussion of such reflection attacks can be found in [35].

3.1. Methods of Attack Amplification

The simplest such attack was the "smurf" attack [22], 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 a TCP amplification attack known as "bang.c". The attacker sends a spoofed TCP SYN with the source address of the victim to an arbitrary TCP server. The server will respond with a SYN|ACK that 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.

Care must also be taken with services that relay requests. If an attacker can send a request to a proxy, and that proxy now attempts to connect to a victim whose address is chosen by the attacker, then, if the proxy repeatedly resends the request when receiving no answer, this can also serve as an attack amplifier.

Another variant of amplification occurs in protocols that include, within the protocol payload, an IP address or name of host to which subsequent messages should be sent. An example of such a protocol is the Session Initiation Protocol (SIP) [50], which carries a payload defined by the Session Description Protocol (SDP) [51]. The SDP payload of the SIP message conveys the IP address and port to which media packets, typically encoded using the Real Time Transport Protocol (RTP) [52], are sent.

To launch this attack, an attacker sends a protocol message, and sets the IP address within the payload to point to the attack target. The recipient of the message will generate subsequent traffic to that IP address. Depending on the protocol, this attack can provide substantial amplification properties. In the specific case of SIP, if a caller makes calls to high-bandwidth media sources (such as a video server or streaming audio server), a single SIP INVITE packet, typically a few hundred bytes, can result in a nearly continuous stream of media packets at rates anywhere from a few kbits per second up to megabits per second. This particular attack is called the "voice hammer".

Unlike the other techniques described above, this technique does not require the attacker to modify packets or even spoof their source IP address. This makes it easier to launch.

This attack is prevented through careful protocol design. Protocols should, whenever possible, avoid including IP addresses or hostnames within protocol payloads as addresses to which subsequent messaging should be sent. Rather, when possible, messages should be sent to the source IP from which the protocol packet came. If such a design is not possible, the protocol should include a handshake whereby it can be positively determined that the protocol entity at that IP address or hostname does, in fact, wish to receive that subsequent messaging. That handshake itself needs to be lightweight (to avoid being the source of another DoS attack), and secured against the spoofing of the handshake response.

Finally, a somewhat similar attack is possible with some protocols where one message leads to another message that is not sent as a reply to the source address of the first message. This can be an

issue with protocols to enable mobility, for example, and might permit an attacker to avoid ingress filtering. Such protocols are notoriously difficult to get right.

3.2. Strategies to Mitigate Attack Amplification

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

- As far as possible, perform ingress filtering [7] [39] to prevent source address spoofing.
- 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.
- Proxies should not arbitrarily relay requests to destinations chosen by a client.
- Avoid signaling third-party connections. Any unavoidable third-party connections set up by a signaling protocol should incorporate lightweight validation before sending significant data.

4. DoS Mitigation Strategies

A general problem with DoS defense is that it is not in principle possible to distinguish between a flash crowd and a DoS attack. Indeed, having your site taken down by a flash crowd is probably a more common experience than having it DoS-ed -- so common it has acquired its own names: being Slashdotted or Farked, after the web sites that are common sources of flash crowds. Thus, the first line of defense against DoS attacks must be to provision your service so that it can handle a foreseeable legitimate peak load. Underprovisioned sites are the easiest to take down.

Specific strategies for DoS defense fall into two broad categories:

1. Avoiding allowing attacks that are better than generic resource consumption.
2. Minimizing the extent to which generic resource consumption attacks crowd out legitimate users.

In the remainder of this section, we consider specific applications of these two approaches at a variety of levels of network system architecture.

4.1. Protocol Design

4.1.1. 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 [2] [14]. 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.

4.1.2. Make It Hard to Simulate a Legitimate User

Other than having massive overcapacity, the only real defense against resource consumption attacks is to preferentially discriminate against attackers. The general idea is to find something that legitimate users can do but attackers can't. The most commonly proposed approaches include:

1. Puzzles: force the attacker to do some computation that would not be onerous for a single user but is too expensive to do en masse [14].

2. Reverse Turing tests: specialized puzzles that are hard for machines to do but easy for humans, thus making automated attacks hard [13].

3. Reachability testing: force the proposed client to demonstrate that it can receive traffic at a given IP address. This makes it easier to trace attackers.

All of these techniques have substantial limitations. Puzzles tend to discriminate against legitimate users with slow computers. In addition, the wide availability of remotely controlled compromised machines ("bots") means that attackers have ample computing power at their disposal. There has been substantial work in attacking reverse Turing tests automatically, thus making them of limited applicability. Finally, reachability testing is substantially weakened by bots because the attacker does not need to hide his source address.

4.1.3. 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 [53] 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 [3] is hard to use in the presence of overload. A BGP router that cannot store a route it received has two choices: completely restart BGP or shut down one or more peerings [26]. 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.

4.1.4. 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, especially because there are environments in which it is desirable for units to operate with effectively no authentication (e.g., airport hotspots). 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. However, this also implies that the network elements should avoid acting for unauthenticated hosts, instead just letting them access the network more or less directly.

4.2. Network Design and Configuration

In general, networks should be provisioned with private, out-of-band access to console or control ports so that such control facilities will be available in the face of a DoS attack launched against either the control or data plane of the (in-band) network. Typically, such out-of-band networks are provisioned on a separate infrastructure for exactly this purpose. Out-of-band access is a crucial capability for DoS mitigation, since many of the typical redundancy and capacity management techniques (such as prioritizing routing or network management traffic) fail during such attacks. In addition, many redundancy protocols such as VRRP [47] can fail during such attacks as they may be unable to keep adjacencies alive.

There are several default configuration settings that can also be exploited to generate several of the attacks outlined in this document. For example, some vendors may have features such as IP redirect, directed broadcast, and proxy ARP enabled by default. Similar defaults, such as publicly readable SNMP [48] communities (e.g., "public") can be used to reveal otherwise confidential information to a prospective attacker. Finally, other unauthenticated configuration management protocols such as TFTP [49] should be avoided if possible; at the very least access to TFTP configuration archives should be protected and TFTP should be filtered at administrative boundaries. Finally, since many of the password encryption techniques used by router vendors are reversible, keeping such passwords on a configuration archive (as part of a configuration file), even in the encrypted form written by the router, can lead to unauthorized access if the archive is compromised.

4.2.1. Redundancy and Distributed Service

A basic principle of designing systems to handle failure is to have redundant servers that can take over when one fails. This is equally true in the case of DoS attacks, which often focus on a given server and/or link. If service delivery points can be distributed across the network, then it becomes much harder to attack the entire service. In particular, this makes attacks on a single network link more difficult.

4.2.2. 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 [9] or IPsec authentication, combined with the GTSM [8] to prevent eBGP association with non-immediate neighbors. In the future, this will likely imply better authentication of the routing information itself.

4.2.3. 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.

4.3. Router Implementation Issues

Because a router can be considered as an end-system, it can potentially benefit from all the prevention mechanisms prescribed for end-system implementation. However, one basic distinction between a router and a host is that the former implements routing protocols and forwards data, which in turn lead to additional router-specific implementation considerations. The issues described below are meant to be illustrative and not exhaustive.

4.3.1. Checking Protocol Syntax and Semantics

Protocol syntax defines the formation of the protocol messages and the rules of exchanges. The questions addressed by protocol syntax checking includes, but is not limited to, the following:

1. Who sent the message?

2. Does the content conform to the protocol format?

3. Was the message sent with correct timing?

The first step in protocol syntax verification is to ensure that an incoming message was sent by a legitimate party. There are multiple ways to perform this check. One can verify the source IP address and even the MAC address of the message. Utilizing the fact that eBGP peers are normally directly connected, one can also check the TTL value in a packet and discard any BGP updates packet whose TTL is less than some maximum value (typically, max TTL - 1) [8]. Cryptographic authentication should also be used whenever available to verify that an incoming message is indeed from an expected sender. For BGP, at the very least, this implies the use of TCP MD5 [9] or IPsec authentication.

In addition to the sender verification, it is also important to check the syntax of a received routing message, as opposed to assuming that all messages came in a correct format. It happened in the past that routers crashed upon receiving ill-formed routing messages. Such faults will be prevented by performing rigorous syntax checking.

4.3.2. Consistency Checks

Protocol semantics define the meaning of the message content, the interpretation of the values, and the actions to be taken according to the content. Here is a simple example of using semantic checking. When a link failure causes a router in Autonomous System (AS) A to send a peer router B a withdrawal message for prefix P, B should make sure that any alternative path it finds to reach P does not go through A. This simple check is shown to significantly improve BGP convergence time in many cases [42].

Another example of using semantic checking against false routing injection is described in [44]. The basic idea is to attach to the route announcement for prefix P a list of the valid origin ASes. Due to the rich connectivity in today's Internet topology, a remote AS will receive routing updates from multiple different paths and can check to see whether each update carries the identical origin AS list. Although a false origin may announce reachability to P, or alter the origin AS list, it would be difficult, if not impossible, to block the correct updates from propagating out, and thus remote ASes can detect the existence of false updates by observing the inconsistency of the received origin AS lists for P. Research studies show that the "allowed origin list" test can effectively detect the majority of falsely originated updates.

Generally speaking, verifying the validity of BGP routes can be challenging because BGP is policy driven and policies of individual ISPs are not known in most cases. But assuming that policies do not change in short time scale, in principle one could verify new updates against observed routes from the recent past, which reflect the

routing policies in place. Research work is needed to explore this direction.

Note that while the above steps are all fairly simple and don't really "bulletproof" the protocol, each adds some degree of protection. As such, the combination of the above techniques can result in an effective reduction in the probability of undetected faults.

4.3.3. Enhance Router Robustness through Operational Adjustments

There exist a number of configuration tunings that can enhance robustness of BGP operations. One example is to let BGP peers coordinate the setting of a limit on the number of prefixes that one BGP speaker will send to its peer [43]. Although such a check does not validate the prefix owned by each peer, it can prevent false announcements of large numbers of invalid routes. Had all BGP routers been configured with this simple checking earlier, several large-scale routing outages in the past could have been prevented. Note, however, that care must be taken with hard limits of this type because they can be used to mount a DoS because implementations often discard excess routes indiscriminately, thus potentially causing black-holing of correct routes.

Another example of useful configuration tuning is to adjust the BGP's KeepAlive and Hold Timer values to minimize BGP peering session resets. Previous measurements show that heavy traffic load, such as those caused by worms, can cause BGP KeepAlive messages to be delayed or dropped, which in turn cause BGP peering session breakdown. Such load-induced session breaks and re-establishments can lead to an excessive amount of BGP updates during the periods when stable routing is needed most.

4.3.4. Proper Handling of Router Resource Exhaustion

In addition to the resource exhaustion problems that are generally apply to all end-systems, as described in Section 2, router implementations must also take special care in handling resource exhaustions when they occur in order to keep the router operating despite the problem. For example, under normal operations a router does not require a large cache to hold outstanding ARP requests because the replies are normally received within a few milliseconds. However, certain conditions can lead to ARP cache exhaustion, for example, during a virus attack where many packets are sent to non-existing IP addresses, thus there are no ARP replies to the requests for those addresses. Such phenomena have happened in the past and led to routers failing to forward packets.

Another example is queue management. Many high-end routers are designed so that most packets can be handled purely in specialized processors (Application-Specific Integrated Circuit (ASICs), Field Programmable Gate-Arrays (FPGAs), etc.). However, exceptional packets must be routed to a supporting general purpose CPU for handling. On some such systems, it may be possible mount a low-effort DoS attack by saturating the queues between the specialized hardware and the supporting processor.

So the attack vector on routers/network devices is a low packets-per-second (PPS) queue saturation attack on the ASIC's raw queue structure. The countermeasure here is to have multiple such queues designed in such a way that it's difficult for an attacker to arrange to fill multiple queues [45].

4.4. End-System Implementation Issues

4.4.1. State Lookup Complexity

Any system that instantiates per-connection state should take great care to implement state-lookup mechanisms in such a way that performance cannot 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.

4.4.1.1. 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 [33], 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 than 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.

4.4.1.2. 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 communication 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 be randomly allocated.

With DNS, the ID that 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.

4.4.2. Operational Issues

4.4.2.1. Eliminate Bad Traffic Early

Many DoS attacks are generic bandwidth consumption attacks that operate by clogging the link that connects the victim server to the Internet. Filtering these attacks at the server does no good because the traffic has already traversed the link that is the scarce resource. Such flows need to be filtered at some point closer to the attacker. Where possible, operators should filter out obviously bad traffic. In particular, they should perform ingress filtering [7].

4.4.2.2. Establish a Monitoring Framework

Network operators are strongly encouraged to establish a monitoring framework to detect and log abnormal network activity. One cannot defend against an attack that one doesn’t detect or understand. Such monitoring tools can be used to set a baseline of "normal" traffic, and can be used to detect aberrant flows and determine the type and source of the aberrant flows. This is extremely helpful when responding to distributed DoS attacks 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 attacks 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 [24] [39] [25].

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. Security Considerations

This entire document is about security.

7. Acknowledgements

We are very grateful to Vern Paxson, Paul Vixie, Rob Thomas, Dug Song, George Jones, Jari Arkko, Geoff Huston, and Barry Greene for their constructive comments on earlier versions of this document.

8. Normative References

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

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

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

[4] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[5] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[6] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[7] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[8] Gill, V., Heasley, J., and D. Meyer, "The Generalized TTL Security Mechanism (GTSM)", RFC 3682, February 2004.

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

[10] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

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

[12] Waitzman, D., Partridge, C., and S. Deering, "Distance Vector Multicast Routing Protocol", RFC 1075, November 1988.

[13] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In Proceedings of Eurocrypt, 2003.

9. Informative References

[14] 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.

[15] 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.

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

[17] 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.

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

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

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

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

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

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

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

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

[26] 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.

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

[28] 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.

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

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

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

[32] Fenner, B., Ed., and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, October 2003.

[33] 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.

[34] Watson, P., "Slipping in the Window: TCP Reset attacks", Presentation at 2004 CanSecWest, http://www.cansecwest.com/archives.html.

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

[36] Joe Stewart, "DNS Cache Poisoning - The Next Generation", Jan 27 2003, http://www.lurhq.com/dnscache.pdf.

[37] Stewart, R., Ed., and M. Dalal, Ed., "Improving TCP's Robustness to Blind In-Window Attacks", Work in Progress, June 2006.

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

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

[40] D. Wessels, "Running An Authoritative-Only BIND Nameserver", http://www.isc.org/index.pl?/pubs/tn/index.pl?tn=isc-tn-2002-2.txt.

[41] M. Zalewski, "Strange Attractors and TCP/IP Sequence Number Analysis", http://www.bindview.com/Services/Razor/Papers/2001/tcpseq.cfm.

[42] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, F. S. Wu, and L. Zhang. Improving BGP Conver-gence Through Assertions Approach. In Proc. of IEEE INFOCOM, June 2002.

[43] Chavali, S., Radoaca, V., Miri, M., Fang, L., and S. Hares, "Peer Prefix Limits Exchange in BGP", Work in Progress, April 2004.

[44] X. Zhao, D. Massey, A. Mankin, S.F. Wu, D. Pei, L. Wang, L. Zhang, "BGP Multiple Origin AS (MOAS) Conflicts", http://nanog.org/mtg-0110/lixia.html, 2001.

[45] Cisco Systems, "Building Security Into the Hardware", ftp://ftp-eng.cisco.com/cons/isp/security/CPN-Summit-2004/Paris-Sept-04/SE14-BUILDING-SECURITY-INTO-THE-HARDWARE-c1_8_30_04.pdf, 2004.

[46] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[47] Hinden, R., "Virtual Router Redundancy Protocol (VRRP)", RFC 3768, April 2004.

[48] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[49] Malkin, G. and A. Harkin, "TFTP Timeout Interval and Transfer Size Options", RFC 2349, May 1998.

[50] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[51] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[52] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[53] Hedrick, C., "Routing Information Protocol", RFC 1058, June 1988.

Appendix A. IAB Members at the Time of This Writing

- Bernard Aboba
- Loa Andersson
- Brian Carpenter
- Leslie Daigle
- Elwyn Davies
- Kevin Fall
- Olaf Kolkman
- Kurtis Lindvist
- David Meyer
- David Oran
- Eric Rescorla
- Dave Thaler
- Lixia Zhang

Authors' Addresses

Mark J. Handley, Ed.
UCL
Gower Street
London WC1E 6BT
UK

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

Eric Rescorla, Ed.
Network Resonance
2483 E. Bayshore #212
Palo Alto 94303
USA

EMail: ekr@networkresonance.com

Internet Architecture Board
IAB

EMail: iab@ietf.org

Full Copyright Statement

Copyright (C) The IETF Trust (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST, AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.