Estonia is one of the most heavily computerized countries in the world and has pioneered e-government; the slowdown on major government and commercial sites for almost a month had a serious impact on their citizens' ability to do business.

The Estonian computer emergency response team determined that the attacks were coming largely from outside Estonia. Experts acted quickly to close down sites under attack and to apply other controls to limit inbound traffic. Emergency response teams from the European Union and the United States were called in to help manage the attack [VAM07].

Pinpointing the source of the attack was not possible, The source of such attacks is often unclear, because determining where the traffic was routed from most recently is not the same is identifying the original source of the attack. Although the Estonian Foreign Minister accused the Kremlin of involvement, the Defense Minister acknowledged there was no definitive evidence of that. One Russian was convicted in Estonia of a minor role in the attack. Responsibility for planning, coordinating, and mounting the attack has not been, and probably never will be established [EVR09].

THREAT: DENIAL OF SERVICE

A **denial of service**, or **DoS**, attack is an attempt to defeat availability, the third of the three basic properties to be preserved in computer security. Denial of service means just what its name implies: a user is denied access to authorized services or data. Confidentiality and integrity are concerned with preventing unauthorized access; availability is concerned with preserving authorized access.

Confidentiality and integrity tend to be binary: Data or objects either are or are not kept private and unmodified. Availability can be more nuanced, in that there may be service but in insufficient quantity or at unacceptable responsiveness. You know that a web page takes a few seconds to load, but as time passes you become more frustrated or suspicious that it will never display; then, suddenly it appears and you wonder why it took so long. Thus, denial of service ranges from complete loss of access to noticeable and unacceptable slowing to inconvenience.

In this chapter we describe what causes denial of service. Many causes are nonmalicious and often sporadic and spontaneous, so little can be done about them. We focus on the malicious causes because those are the ones that can be dealt with. Fortunately, several classes of countermeasures are effective against malicious denial-of-service attacks. First, we consider some of the causes.

THREAT: FLOODING

Imagine a teacher in a classroom full of six-year-olds. Each child demands the teacher's attention. At first, the teacher hears one child and gives the child attention. Then a second child calls, and the teacher focuses on that child while trying to remember what the first child needed. Seeing that calling out works, children three, four, and

five cry out for the teacher, but this frustrates other children who also seek attention. Of course, each child who calls out does so more loudly than the previous ones, and soon the classroom is a cacophony of children's shouts, making it impossible for the teacher to do anything except tell them all to be quiet, wait their turn, and be patient (none of which comes naturally to six-year-olds). The teacher becomes so overloaded with demands that the only solution is to dismiss all current demands and start afresh.

An attacker can try for the same overloading effect by presenting commands more quickly than a server can handle them; servers often queue unmet commands during moments of overload for service when the peak subsides, but if the commands continue to come too quickly, the server eventually runs out of space to store the demand. Such an attack is called an **overload** or **flood**.

The target of a flooding attack can be an application, such as a database management system; an operating system or one of its components, for example, a file or print server; or a network appliance like a router. Alternatively, the flooding attack can be directed against a resource, such as a memory allocation table or a web page. On the day Michael Jackson died, Google received so many queries about him that the Google engineers thought they were under attack and took evasive measures that, ironically, limited access to the Google news service. A denial-of-service flooding attack can be termed **volumetric**, meaning it simply seeks to saturate or exhaust the capacity of a critical telecommunications link.

THREAT: BLOCKED ACCESS

As another physical analogy, consider a traffic accident that stops traffic in both directions of a busy, two-lane road. As motorists begin to line up behind the accident, at some point one driver concludes the right approach is to slip into the oncoming traffic lane to get around all the stopped cars and, naturally, others immediately follow. They get as far as the accident and have to stop. What then happens is that two lanes of traffic build up at the point of the accident on both sides of the accident, meaning that police and other emergency vehicles cannot get past the two solid lines of cars in both directions to get to the accident. Even when the disabled cars are pushed off the road to clear the accident, all lanes are filled with cars that cannot move because there is no room either in front or behind.

In computer security, the attacker may simply prevent a service from functioning. The attacker could exploit a software vulnerability in an application and cause the application to crash. Or the attacker could interfere with the network routing mechanisms, preventing access requests from getting to the server. Yet another approach would be for the attacker to manipulate access control data, deleting access permissions for the resource, or to disable the access control mechanism so that nobody could be approved for access. In Sidebar 15-1, the attacker alleged that he had deleted the original copy of an important database and encrypted a backup copy, which he was holding for ransom.

State of Virginia Database Held for Ransom **Sidebar 15-1**

State officials in Virginia received a ransom note in May 2009 demanding $10 million for release of a state database of 8.3 million records of drug prescriptions for state residents. The database held copies of prescriptions for federal controlled substances filled since 2003.

Ransom note:

> ATTENTION VIRGINIA
> I have your s[censored]! In *my* possession, right now, are 8,257,378 patient records and a total of 35,548,087 prescriptions. Also, I made an encrypted backup and deleted the original. Unfortunately for Virginia, their backups seem to have gone missing, too. Uhoh:
> For $10 million, I will gladly send along the password. You have 7 days to decide. If by the end of 7 days, you decide not to pony up, I'll go ahead and put this baby out on the market and accept the highest bid. Now I don't know what all this [censored] is worth or who would pay for it, but I'm bettin' someone will. Hell, if I can't move the prescription data at the very least I can find a buyer for the personal data (name, age, address, social security #, driver's license #). (Brian Krebs, *Washington Post* Security Fix blog, May 4, 2009)

Although the attacker alleged that he had deleted the original, made one encrypted backup copy, and deleted all other backups, state officials were able to restore the database from backup copies and could access it with no difficulty. Sandra Whitley Ryals, director of the Virginia Department of Health Professions stated, "We are satisfied that all data was properly backed up and that these backup files have been secured" (WHSV TV, Richmond, VA, from WHSV.com, May 6, 2009). Thus, the ransom demand seems to have been a hoax. Nevertheless, removing sensitive data and holding it for ransom is a potentially effective means to block access.

THREAT: ACCESS FAILURE

Either maliciously or not, hardware and software fail from time to time; of course, it always seems that such nonmalicious failures occur only at critical times. Software stops working due to a flaw, or a hardware device wears out or inexplicably stops. The failure can be sporadic, meaning that it goes away or corrects itself spontaneously, or the failure can be permanent, as from a faulty component.

These, then, are the three root threats to availability:

- insufficient capacity; overload
- blocked access
- unresponsive component

The attacker will try to actualize any of these threat types by exploiting vulnerabilities against them. In the next section we examine some of these potential vulnerabilities. In the following case we describe an incident that resulted from a combination of factors—none malicious—including age of components, network design, and communications protocols.

causes, no one of which is enough by itself to cause a problem, can interact in a way that becomes serious. Yet teasing out the individual causes can be challenging to an administrator, especially when faced with the immediate problem of trying to get a failed system operating again.

From the three basic causes of failed service (lack of capacity or overload, blocked access, and unresponsive components), we move now to identify the vulnerabilities that could lead to these failures.

VULNERABILITY: INSUFFICIENT RESOURCES

In our example of the teacher and the six-year-olds, the teacher simply could not handle demands from all the students: one at a time, perhaps, but not all at once. One teacher with two or three students could probably have coped, or ten teachers with thirty students, but not one against thirty. Similarly with computing systems, the attacker can try to consume a critical amount of a scarce resource.

Flooding a victim is basically an unsophisticated attack, although the means of performing the flooding can become sophisticated. Another way to deny service is to block access to a resource, which we consider next.

Insufficient Capacity

If the attacker has greater bandwidth than the victim, the attacker can overwhelm the victim with the asymmetry. A victim is always potentially vulnerable to an attacker with more resources. Examples of insufficient resources may be slots in a table of network connections, room in a buffer, or cycles of a processor.

Denial of service is especially noticeable in network attacks, in which the attacker can consume too much of the available network bandwidth. We consider network capacity exhaustion next.

Network Flooding Attack

The most primitive denial-of-service attack is flooding a connection. If an attacker sends you as much data as your communications system can handle, you are prevented from receiving any other data. Even if an occasional packet reaches you from someone else, communication to you will be seriously degraded. Ironically, this problem is exacerbated by the robustness of the TCP protocol: If, because of congestion, packets 1 and 2 are delayed but packet 3 manages to slip through first, the protocol handler will notice that 1 and 2 are missing. The receiver accepts and holds packet 3, but the sender may retransmit packets 1 and 2, which adds to the congestion.

More sophisticated attacks use elements of Internet protocols. In addition to TCP and UDP, there is a third class of protocols, called ICMP or Internet Control Message Protocols. Normally used for system diagnostics, these protocols do not have associated user applications. ICMP protocols include

- *ping*, which requests a destination to return a reply, intended to show that the destination system is reachable and functioning

- *echo*, which requests a destination to return the data sent to it, intended to show that the connection link is reliable (ping is actually a version of echo)
- *destination unreachable*, which indicates that a destination address cannot be accessed
- *source quench*, which means that the destination is becoming saturated and the source should suspend sending packets for a while

These protocols have important uses for network management. But they can also be used to attack a system. The protocols are handled within the network stack, so the attacks may be difficult to detect or block on the receiving host. We examine how these protocols can be used to attack a victim.

Ping of Death

A **ping of death** is a simple attack, using the ping command that is ordinarily used to test response time from a host. Since ping requires the recipient to respond to the packet, all the attacker needs to do is send a flood of pings to the intended victim. The attack is limited by the smallest bandwidth on the attack route, as shown in Figure 15-1. If the attacker is on a 10-megabyte (MB) connection and the path to the victim is 100 MB or more, mathematically the attacker alone cannot flood the victim. But the attack succeeds if the numbers are reversed: The attacker on a 100-MB connection can certainly flood a 10-MB victim. The ping packets will saturate the victim's bandwidth.

Smurf

The **smurf** attack is a variation of a ping attack. It uses the same vehicle, a ping packet, with two extra twists. First, the attacker chooses a network of unwitting victims that

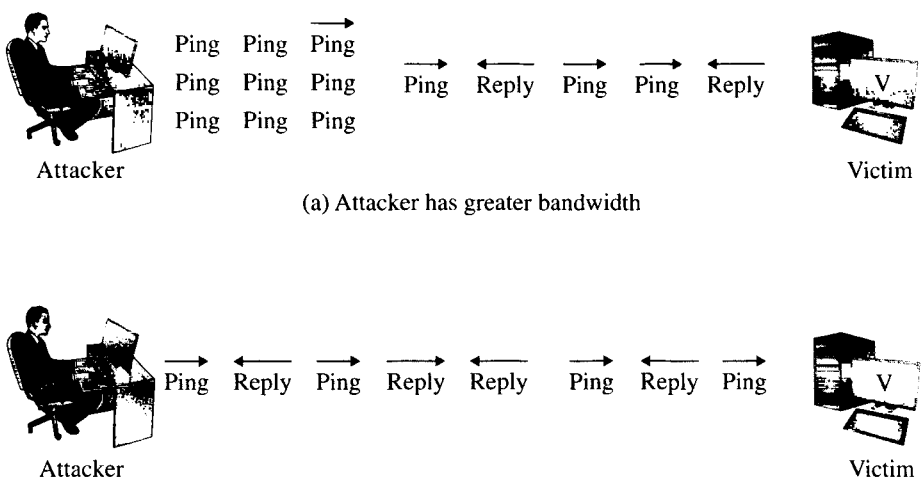

FIGURE 15-1 Ping Attack: (a) Attacker Has Greater Bandwidth; (b) Victim Has Greater Bandwidth

become accomplices. The attacker spoofs the source address in the ping packet so that it appears to come from the victim, which means a recipient will respond to the victim. Then, the attacker sends this request to the network in broadcast mode by setting the last byte of the address to all 1s; broadcast mode packets are distributed to all hosts on the subnetwork. The attack is depicted in Figure 15-2, showing the single broadcast attack being reflected back on the victim. In this way the attacker uses the entire subnetwork to multiply the attack's effect.

Echo–Chargen

The **echo–chargen** attack works between two hosts. Chargen is an ICMP protocol that generates a stream of packets to test the network's capacity. Echo is another ICMP protocol used for testing; a host receiving an echo returns everything it receives to the sender.

The attacker picks two victims, A and B, and then sets up a chargen process on host A that generates its packets as echo packets with a destination of host B. Thus, A floods B with echo packets. But because these packets request the recipient to echo them back to the sender, host B replies by returning them to host A. As shown in Figure 15-3, this series puts the network infrastructures of A and B into an endless loop, as A generates a string of echoes that B dutifully returns to A, just as in a game of tennis. Alternatively, the attacker can make B both the source and destination address of the first packet, so B hangs in a loop, constantly creating and replying to its own messages.

SYN Flood

Another popular denial-of-service attack is **the SYN flood**. This attack uses the TCP protocol suite, making the session-oriented nature of these protocols work against the victim.

For a protocol such as Telnet or SMTP, the protocol peers establish a virtual connection, called a session, to synchronize the back-and-forth, command–response nature of the interaction. A session is established with a three-way TCP handshake. Each TCP packet has flag bits, two of which are denoted SYN (synchronize) and ACK (acknowledge). First, to initiate a TCP connection, the originator sends a packet with the SYN bit on. Second, if the recipient is ready to establish a connection, it replies

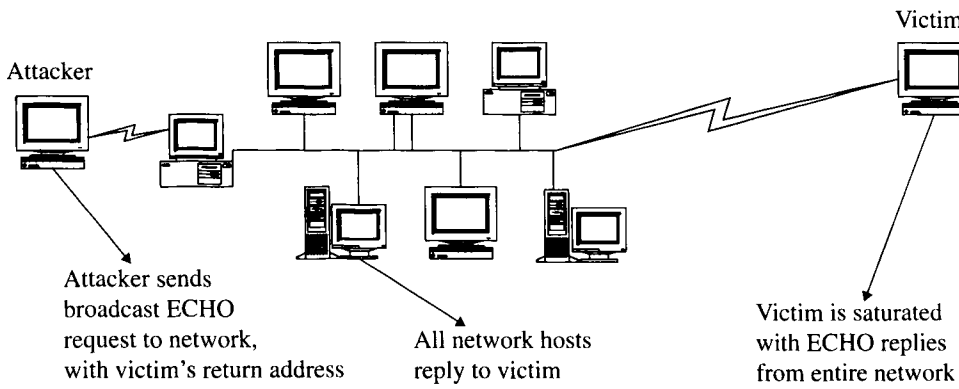

FIGURE 15-2 Smurf Attack

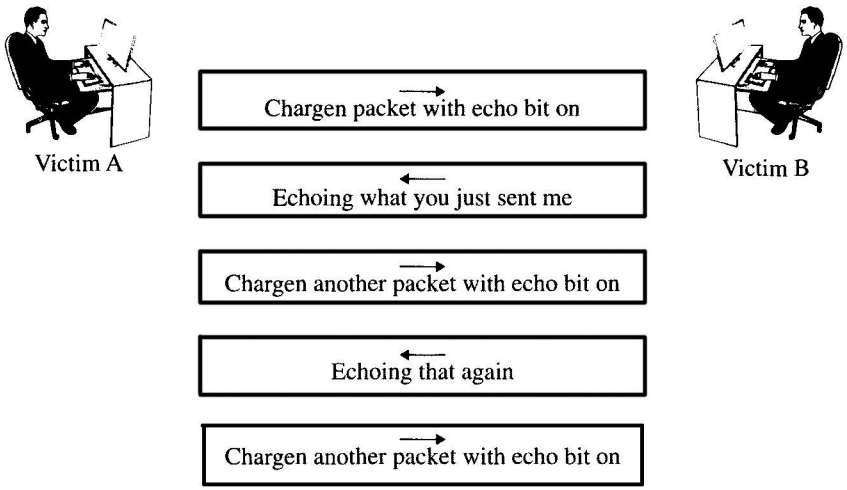

FIGURE 15-3 Echo–Chargen Attack

with a packet with both the SYN and ACK bits on. Finally, the first party completes the exchange to demonstrate a clear and complete communication channel by sending a packet with the ACK bit on, as shown in Figure 15-4.

Occasionally packets get lost or damaged in transmission. The destination (which we call the recipient) maintains a queue called the SYN_RECV connections, tracking those items for which a SYN–ACK has been sent but no corresponding ACK has yet been received. Normally, these connections are completed in a short time. If the SYN–ACK (2) or the ACK (3) packet is lost, eventually the destination host will time out the incomplete connection and discard it from its waiting queue.

The attacker can deny service to the target by sending many SYN requests, to which the target properly responds with SYN–ACK; however, the attacker never replies with ACKs to complete the connections, thereby filling the victim's SYN_RECV queue. Typically, the SYN_RECV queue is quite small, holding 10 or 20 entries. Because of potential routing delays in the Internet, typical holding times for the SYN_RECV queue can be minutes. So the attacker need only send a new SYN request every few seconds, and the queue will fill.

Attackers using this approach usually do one more thing: They spoof a nonexistent return address in the initial SYN packet. Why? For two reasons. First, the attacker does not want to disclose the real source address in case someone should inspect the packets in the SYN_RECV queue to try to identify the attacker. Second, the attacker wants to

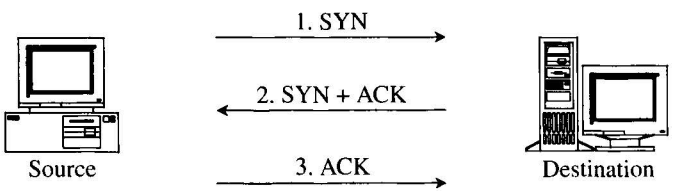

FIGURE 15-4 Three-way TCP Handshake

make the SYN packets indistinguishable from legitimate SYN packets to establish real connections. Choosing a different (spoofed) source address for each one makes them unique, as ordinary traffic would be. A SYN–ACK packet to a nonexistent address results in an ICMP Destination Unreachable response, but this is not the ACK for which the TCP connection is waiting. (TCP and ICMP are different protocol suites, so an ICMP reply does not necessarily get back to the sender's TCP handler.)

These attacks misuse legitimate features of network protocols to overwhelm the victim, but the features cannot be disabled because they have necessary purposes within the protocol suite. Overwhelming network capacity is not the only way to deny service, however. In the next section we examine attacks that exhaust other available resources.

Resource Starvation

Most computer resources, such as memory, disk space, or buffer size, are finite; when allocation of these resources is near the limit, computers can behave badly.

Resource Exhaustion

A computer supports multiple applications by dividing time between applications; operating systems research has helped people design effective algorithms for deciding how much (what proportion of) processing time to allocate to which applications. Switching from one application to another, called **context switching**, requires time and memory because the current state of the application is saved and the previous state of the next application is reloaded. Register values must be written to memory, outstanding asynchronous activities must be completed, dropped, or recorded, and memory must be preserved or freed. If there are few active processes and few context switches, the overhead for each switch is negligible, but as the number of active processes increases, the proportion of time spent in context switching also grows, which means the proportion of time for actual computing decreases. With too many processes, a system can enter a state called **thrashing**, in which its performance fails because of nearly continuous context switching.

Time is not the only resource that can be exhausted. Buffers for incoming email can be overwhelmed by a sudden flood of incoming messages. Logging and log files can be swamped by a large number of errors or fault conditions that must be handled. Buffers for reassembling fragmented communications can also be exhausted.

Even identification and authentication can become vulnerable in an exhaustion attack. To protect against automated guessing attacks, some authentication services temporarily or permanently disable account access after some number, such as three or five, of failed login attempts. Thus, a malicious user can block access by repeatedly failing to log in as the victim.

IP Fragmentation: Teardrop

The **teardrop** attack misuses a feature ironically intended to improve network communication. A network IP datagram is a variable-length object. To support different applications and conditions, the datagram protocol permits a single data unit to be fragmented, that is, broken into pieces and transmitted separately. Each fragment

Routers use complex algorithms to decide how to route traffic. No matter the algorithm, they essentially seek the best path (where "best" is measured in some combination of distance, time, cost, quality, and the like). Routers are aware only of the routers with which they share a direct network connection, and they use gateway protocols to share information about their capabilities. Each router advises its neighbors about how well it can reach other network addresses. This characteristic allows an attacker to disrupt the network.

To see how, keep in mind that in spite of its sophistication, a router is simply a computer with two or more network interfaces. Suppose a router advertises to its neighbors that it has the best path to every other address in the whole network. Soon all routers will direct all traffic to that one router. The one router may become flooded, or it may simply drop much of its traffic. In either case, a lot of traffic never makes it to the intended destination.

DNS Attacks

Our final denial-of-service attack is actually a class of attacks based on the concept of a domain name server. As described in Chapter 14, a domain name server (DNS) is a process that uses and manages a table to convert domain names like att.com into network addresses like 211.217.74.130; this process is called resolving the domain name. A domain name server queries other name servers to resolve domain names it does not know. For efficiency, it caches the answers it receives so that it can convert that name more rapidly in the future. An address mapped by a DNS server can be retained for weeks or months.

In the most common implementations of Unix, name servers run software called Berkeley Internet Name Domain, or BIND, or "named" (a shorthand for "name daemon"). BIND has had numerous flaws, including the now-familiar buffer overflow. By overtaking a name server or causing it to cache spurious entries, an attacker can redirect the routing of any traffic, with an obvious implication for denial of service.

Another way to deny service through address resolution failures involves incapacitating the Internet's DNS system itself. In October 2002, a massive flood of traffic inundated the Internet's top-level domain DNS servers, the servers that form the foundation of the Internet addressing structure. There are 13 top-level domain servers spread around the world; these servers translate the top level, or last part of a network address: the .com, .edu, .fr, .org, or .biz part of a URL. In the 2002 attack, roughly half the flood of traffic came from just 200 addresses. Although some people think the problem was a set of misconfigured firewalls, nobody knows for sure what caused the attack, and even whether it was an attack or an anomalous incident.

Again in 2007, a similar thing happened. On February 6, 2007, the DNS root name servers were hit with two massive denial-of-service attacks for a total of six hours. This time it was clearly an attack, at least part of which originated from the Asia-Pacific region [ICA07]. In this situation also, the impact of the attack was significantly reduced because, between 2002 and 2007, the Internet began using a new design for the root name servers. Called anycast, this technology allows the lookup function to be spread over many computers, even hundreds. Thus, attacks on a single DNS server, or even a small number of servers, have little impact.

For all networks, with or without capable security teams, part of the burden of monitoring and detecting denial-of-service attacks can be handled by software. In the next section we describe intrusion detection and prevention systems, computer devices that do that kind of monitoring.

COUNTERMEASURE: INTRUSION DETECTION AND PREVENTION SYSTEMS

After the perimeter controls, firewall, and authentication and access controls block certain actions, some users are admitted to use a computing system. Most of these controls are preventive: They block known bad things from happening. Many studies (for example, see [DUR99]) have shown that most computer security incidents are caused by insiders, people who would not be blocked by a firewall. And insiders require access with significant privileges to do their daily jobs. The vast majority of harm from insiders is not malicious; it is honest people making honest mistakes. Then, too, there are the potential malicious outsiders who have somehow passed the screens of firewalls and access controls. Prevention, although necessary, is not a complete computer security control; detection during an incident copes with harm that cannot be prevented in advance. Halme and Bauer [HAL95] survey the range of controls to address intrusions.

Intrusion detection systems complement these preventive controls as the next line of defense. An intrusion detection system (IDS) is a device, typically another separate computer, that monitors activity to identify malicious or suspicious events. Kemmerer and Vigna [KEM02] survey the history of IDSs. An IDS is a sensor, like a smoke detector, that raises an alarm if specific things occur. A model of an IDS is shown in Figure 15-6. The components in the figure are the four basic elements of an intrusion detection system, based on the Common Intrusion Detection Framework of [STA96]. An IDS receives raw inputs from sensors. It saves those inputs, analyzes them, and takes some controlling action.

IDSs perform a variety of functions:

- monitoring users and system activity
- auditing system configuration for vulnerabilities and misconfigurations
- assessing the integrity of critical system and data files
- recognizing known attack patterns in system activity
- identifying abnormal activity through statistical analysis
- managing audit trails and highlighting user violation of policy or normal activity
- correcting system configuration errors
- installing and operating traps to record information about intruders

No one IDS performs all of these functions. Let us look more closely at the kinds of IDSs and their use in providing security.

Types of IDSs

The two general types of intrusion detection systems are signature based and heuristic. **Signature-based** intrusion detection systems perform simple pattern-matching and report situations that match a pattern corresponding to a known attack type. **Heuristic**

Adaptive Behavior

Because of these limitations of humans, an IDS can sometimes be configured to take action to block the attack or reduce its impact. Here are some of kinds of actions and IDS can take:

- *Continue to monitor* the network.
- *Block the attack* by redirecting attack traffic to a monitoring host, discarding the traffic, or terminating the session.
- *Reconfigure the network* by bringing other hosts online (to increase capacity) or adjusting load balancers.
- *Adjust performance* to slow the attack, for example, by dropping some of the incoming traffic.
- *Deny access* to particular network hosts or services.
- *Shut down* part of the network.
- *Shut down* the entire network.

Counterattack

A final action that can be taken on a detection of an attack is to mount an offense, to strike back. An example of such an attack is described in Sidebar 15-3. Offensive action must be taken with great caution for several reasons:

- The apparent attacker may not be the real attacker. Determining the true source and sender of Internet traffic is not foolproof. Taking action against the wrong party only makes things worse.
- A counterattack can lead to a real-time battle in which both the defenses and offenses must be implemented with little time to assess the situation.
- Retaliation in anger is not necessarily well thought out.
- Legality can shift. Measured, necessary action to protect one's resources is a well-established legal principle. Taking offensive action opens one to legal jeopardy, comparable to that of the attacker.
- Provoking the attacker can lead to escalation. The attacker can take the counterattack as a challenge.

Honeypots

A **honeypot** is a faux environment intended to lure an attacker. It can be considered an IDS, in the sense that the honeypot may record an intruder's actions and even attempt to trace who the attacker is from actions, packet data, or connections.

Network security administrators have two motivations for installing a honeypot. First, they create an environment readily available and intriguing for the attacker, thereby hoping the attacker will focus on the honeypot system, not the real, sensitive system. This approach, called the low-interaction model, requires effort to build the initial faux environment, but after that, the honeypot largely operates automatically. After-the-fact monitoring can sometimes reveal from where the attacker is operating, although such source data is scarce and not always reliable.

Counter-Counter-Countermeasures? Sidebar 15-3

WikiLeaks, formed in December 2006, is a service that makes public leaked sensitive data by posting the data on its web site. On November 22, 2010 it announced it was going to leak a massive number of internal U.S. diplomatic messages beginning on November 28. On November 28, it announced its web site was under a serious denial-of-service attack, even before the first release of diplomatic messages, but WikiLeaks continued to release the messages.

Unknown people, presumably angered by WikiLeaks' breaching security in releasing these cables, apparently launched a denial-of-service attack against WikiLeaks. The severity of the attack was great enough that on December 2 WikiLeaks' hosting provider, Amazon Web Services, a division of online bookseller Amazon.com, cancelled its contract with WikiLeaks, forcing the site to find a new provider. Next, unknown people launched a denial-of-service attack against the DNS provider serving WikiLeaks, EveryDNS. WikiLeaks switched to a Swiss hosting provider, using a network architecture supported by 14 different DNS providers and over 350 mirror sites [BRA10]. Thus, the anti-WikiLeaks forces and their denial-of-service attack caused WikiLeaks to move content and to arrange hosting contracts abruptly.

Meanwhile, the anti- anti-WikiLeaks forces took action. A leaderless group, named Anonymous, on December 8, 2010 launched a series of denial-of-service attacks of their own, called Operation Payback. The targets were MasterCard, which had been accepting donations to transfer to WikiLeaks but had stopped that practice; Amazon, the web hosting company that cancelled service for WikiLeaks; PayPal, which had also stopped accepting payments for WikiLeaks; and other smaller targets. Anonymous involved a group of about 1,500 activist hackers who were organizing in online forums and chats. The attack disabled MasterCard's online services for about six hours.

John Perry Barlow, cofounder of the Electronic Freedom Foundation (EFF) and Fellow at Harvard University's Berkman Center for Internet and Society, tweeted: "The first serious infowar is now engaged. The field of battle is WikiLeaks. You are the troops."

Robert McGrew and Ray Vaughn [MCG06] used a honeypot in a university setting to study network intrusion activity. They started with the honeypot behind a firewall, where they detected no activity. Then they moved it in front of (that is, unprotected by) the firewall, on an unused but accessible network segment. There were no other machines on the segment, so there was no reason for any system to initiate a connection with any address from that segment. An outsider connected to their first honeypot only 2 hours and 40 minutes after they connected the system to the segment. This first honeypot simulated a Sun Solaris (Unix) system. Later they emulated a Windows XP system, to which someone attempted a connection in 14 minutes! The Solaris honeypot logged 117 attacks over a 7-day period for an average of one attack every 1 hour and 26 minutes. The Windows XP honeypot, over an equal time period, logged 212 attacks, averaging an attack every 48 minutes.

In the second, or high-interaction, model, administrators can study the attacker's activities in real time to learn more about the attacker's objectives, tools, techniques, and weaknesses, and then use this knowledge to defend more effectively. Cliff Stoll [STO88] and Bill Cheswick [CHE90] both employed this second form of honeypot to engage with their separate attackers. In these examples, the researchers engaged with the attacker, supplying real or false results in real time. Stoll, for example, decided to simulate the effect of a slow speed, unreliable connection. This gave Stoll the time to analyze the attacker's commands and make certain files visible to the attacker; if the

attacker performed an action that Stoll was not ready for or did not want to simulate, Stoll simply broke off the communication, as if the unreliable line had failed yet again. Obviously, this kind of honeypot requires a great investment of the administrator's time and mental energy.

With a high-interaction honeypot, McGrew and Vaughn detected and engaged with two attackers. By analyzing one attacker's activity, the researchers identified the hacker by name and email address.

Some security researchers operate honeypots as a way of seeing what the opposition is capable of doing. Virus detection companies put out attractive, poorly protected systems and then check how the systems have been infected: by what means, with what result. This research helps inform further product development.

Intrusion detection systems are powerful devices that can identify instances and types of attacks that would require much human analysis. They are subject to false negative and false positive errors, so some attacks will slip through undetected, while other events will be falsely labeled as attacks. Although quality and precision improve regularly, IDSs still have their limitations. In the next section we examine what to expect ideally from an intrusion detection system.

Goals for Intrusion Detection Systems

The two styles of intrusion detection—pattern matching and heuristic—represent different approaches, each of which has advantages and disadvantages. Actual IDS products often blend the two approaches.

Ideally, an IDS should be fast, simple, and accurate, while at the same time being complete. It should detect all attacks with little performance penalty. An IDS could use some—or all—of the following design approaches:

- Filter on packet headers.
- Filter on packet content.
- Maintain connection state.
- Use complex, multipacket signatures.
- Use a minimal number of signatures with maximum effect.
- Filter in real time, online.
- Hide its presence.
- Use an optimal sliding time window size to match signatures.

Stealth Mode

An IDS is a network device (or, in the case of a host-based IDS, a program running on a network device). Any network device is potentially vulnerable to network attacks. How useful would an IDS be if it itself were deluged with a denial-of-service attack? If an attacker succeeded in logging in to a system within the protected network, wouldn't trying to disable the IDS be the next step?

To counter those problems, most IDSs run in **stealth mode**, whereby an IDS has two network interfaces: one for the network (or network segment) being monitored and the other to generate alerts and perhaps perform other administrative needs. The IDS

These management activities are important countermeasures against denial-of-service attacks. You must remember that not all security controls involve software, and sometimes the most effective way to address a threat is with a human being.

CONCLUSION: DENIAL OF SERVICE

In this first half of the chapter we have presented several ways in which service can be denied: by blocking access, by flooding a network with traffic and exceeding capacity, or by destroying a component critical to the service. We also introduced a new countermeasure, the intrusion detection system, and repeated two other ones from previous chapters, network management and procedural protections. The most important points covered thus far are these:

- Denial of service is a threat to the availability of systems and data.
- The causes of denial of service are flooding, blocked access, and component failure. Flooding is the result of excessive demand.
- Vulnerabilities that can cause denial of service are insufficient resources, resource starvation (usually from an attack that overwhelms what would ordinarily be sufficient resources), routing failures from technical or physical causes, and attacks that exercise other vulnerabilities. Resources can include communications bandwidth, buffer and table space, memory or storage capacity, and processor speed or performance.
- Network engineers can tune a network's performance to counter denials of service. Tuning is most effective for nonmalicious situations, but even for malicious attacks, engineers can adjust a network, for example, to deny service to an address from which an attack seems to be coming.
- Intrusion detection and prevention systems monitor a network's activity. Anomaly-based detection works to identify unusual and unacceptable behavior by gathering data on the normal state of a network and measuring the degree of change from the expected norm. Pattern-matching intrusion detection is similar to virus signature detection in that it monitors network activity to identify matches to known malicious attack types.
- Management activities develop and implement policies and procedures that reduce the impact of a security problem.

In thinking about defending against denial-of-service attacks, you should recognize that a snowstorm can be just as effective as a hacker at denying service to a computer system, and a management technique can be just as effective as an IDS at countering certain such attacks. We summarize denial of service threats, vulnerabilities, and countermeasures in Table 15-1.

Most of this part of the chapter has addressed the denial-of-service attack in a computer network, using technical approaches. As we described in the last section, however, human, nontechnical approaches are also important computer security countermeasures.

TABLE 15-1 Threat–Vulnerability–Countermeasure Chart for Denial of Service

Threat	Consequence	Severity	Ease of Exploitation
Denial of service	Loss of service	Serious	Varies
Flooding	Inability to handle volume; loss of service	Serious	Moderate; tools help
Blocked access	Inability to access; loss of access	Serious	Difficult
Access failure	Inability to access	Serious	Difficult

Vulnerability	Exploitability	Prevalence
Insufficient resources	Easy to moderate; depends on ability to mount large demand	Moderate
Resource starvation	Easy to moderate	Moderate
Routing failure	Moderately difficult	Low
Known vulnerability	High	Very high; many attack tools probe for known vulnerabilities
Physical failure	Low for human attacker; high for natural occurrences	Very high, but sporadic

Countermeasure	Addresses Which Issue	Mitigation Type	Mitigation Effect	Effort
Network administration	Insufficient resources, resource starvation, routing failure	Prevention	Fairly high	Low, but constant attention needed
Intrusion detection systems	Some resource exhaustion, known vulnerabilities	Detection and recovery; sometimes prevention	Fairly high	Low
Preparedness	Resource exhaustion, physical failure	Recovery	High	Low

Before we end the discussion, however, we want to expand on network attacks. An ordinary denial-of-service attack in a network is limited by the amount of network traffic one attacker can generate. As we showed earlier, sometimes there is a disparity between the attacker's bandwidth and that of the victim: An attacker on a low capacity medium is mathematically unable to produce enough traffic to overwhelm a victim on a high-capacity line. The situation changes, however, if the attacker can bring together many partners to swarm the victim at once. We consider this massed form of attack, called a distributed denial-of-service attack, in the next section.