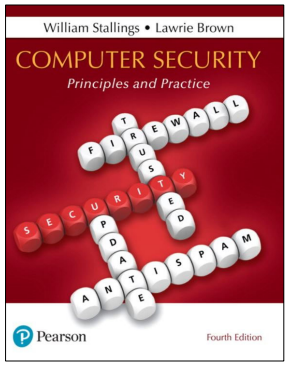

Denial of Service
attacks
CalPolyPomona

Computer Security: Principles and Practice



- **Chapter 7**

- Denial-of-Service Attacks

If this PowerPoint presentation contains mathematical equations, you may need to check that your computer has the following installed:
1) MathType Plugin
2) Math Player (free versions available)
3) NVDA Reader (free versions available)

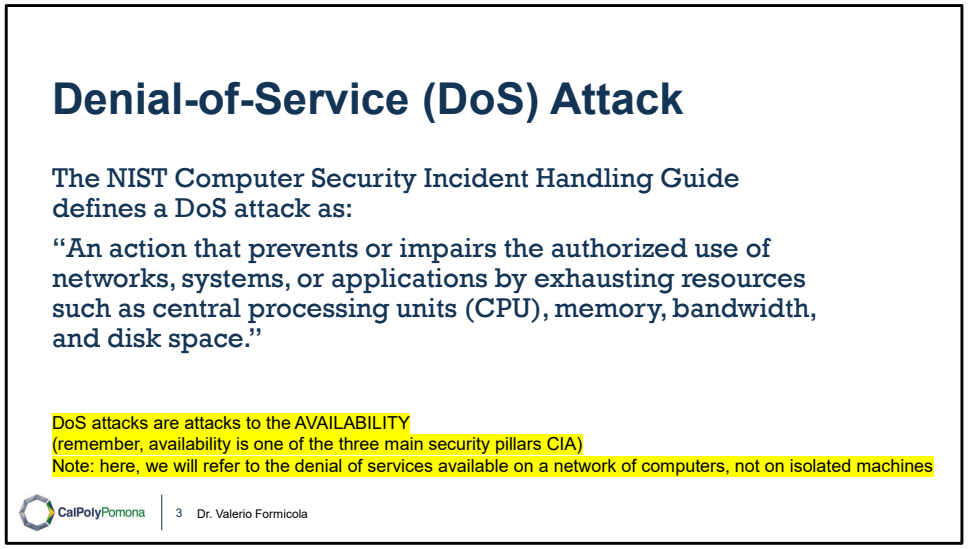
Chapter 1 listed a number of fundamental security services, including **availability.** This service relates to a system being accessible and usable on demand by authorized users. A **denial-of-service attack (DoS)** is an attempt to compromise availability by hindering or blocking completely the provision of some service. The attack attempts to exhaust some critical resource associated with the service. An example is flooding a Web server with so many spurious requests that it is unable to respond to valid requests from users in a timely manner. This chapter explores denial-of-service attacks, their definition, the various forms they take, and defenses against them.

The temporary takedown in December 2010 of a handful of websites that cut ties with controversial website

WikiLeaks, including Visa and MasterCard, made worldwide news. Similar attacks, motivated by a variety of reasons, occur thousands of times each day, thanks in part to the ease by which website disruptions can be accomplished.

Hackers have been carrying out **distributed denial-of-service (DDoS)** attacks for many years, and their potency steadily has increased over time. Due to Internet bandwidth growth, the largest such attacks have increased from a modest 400 Mbps in 2002, to 100 Gbps in 2010 [ARBO10], to 300 Gbps in the Spamhaus attack in 2013, and to 600 Gbps in the BBC attack in 2015. Massive **flooding attacks** in the 50 Gbps range are powerful enough to exceed the bandwidth capacity of almost any intended target, including perhaps the core Internet Exchanges or critical DNS name servers, but even smaller attacks can be surprisingly effective. [SYMA16] notes that DDoS attacks are growing in number and intensity, but that most last for 30 minutes or less, driven by the use of botnets-for-hire. The reasons for attacks include financial extortion, hacktivism, and state-sponsored attacks on opponents. There are also reports of criminals using DDoS attacks on bank systems as a diversion from the real attack on their payment switches or ATM networks. These attacks remain popular as they are simple to setup, difficult to stop, and very effective [SYMA16] .

A DDoS attack in October 2016 represents an ominous new trend in the threat. This attack, on Dyn, a major Domain Name System (DNS) service provider, lasted for many hours and involved multiple waves of attacks from over 100,000 malicious endpoints. The noteworthy feature of this attack is that the attack source recruited IoT (Internet of Things) devices, such as webcams and baby monitors. One estimate of the volume of attack traffic is that it reached a peak as high as 1.2 TBps [LOSH16].

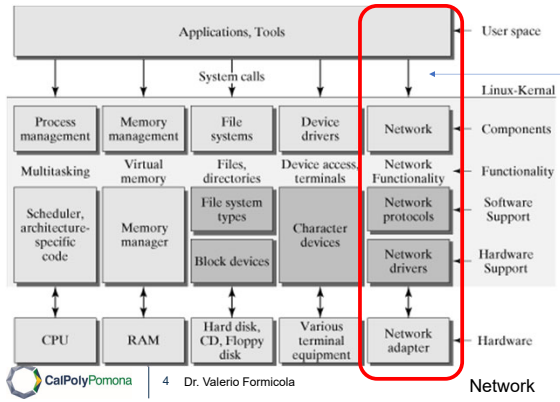


Denial of service is a form of attack on the availability of some service. In the context of computer and communications security, the focus is generally on network services that are attacked over their network connection. We distinguish this form of attack on availability from other attacks, such as the classic acts of god, that cause damage or destruction of IT infrastructure and consequent loss of service.

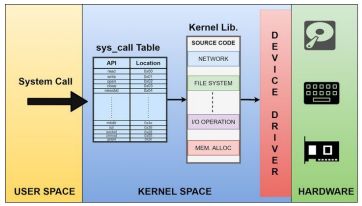
NIST SP 800-61 (*Computer Security Incident Handling Guide* , August 2012) defines denial-of-service (DoS) attack as follows:

A **denial of service (DoS)** is an action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space.

Recall: Operating System – Linux example



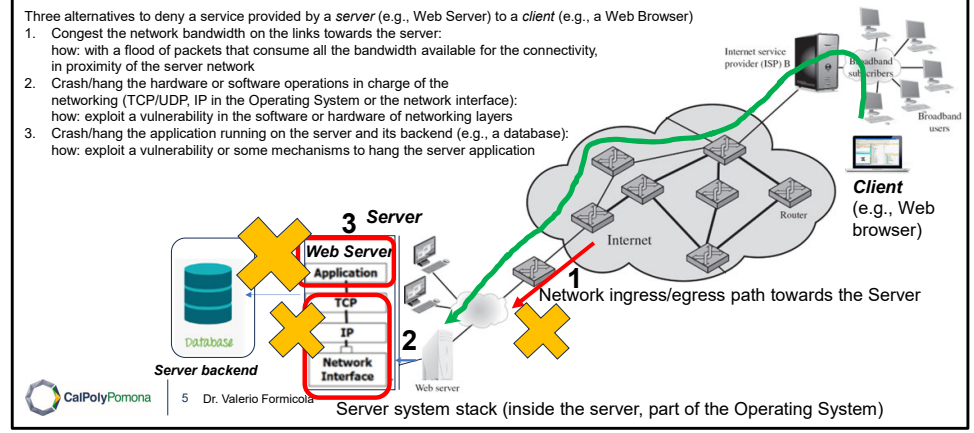
System calls = O.S. "APIs"

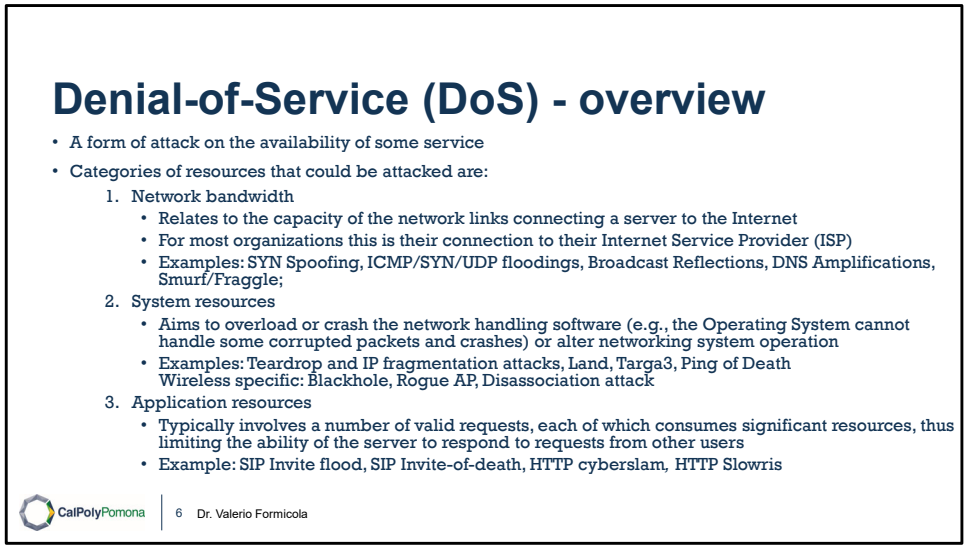


How to stop a client from receiving a service in a remote host

Three alternatives to deny a service provided by a *server* (e.g., Web Server) to a *client* (e.g., a Web Browser)

1. Congest the network bandwidth on the links towards the server:
   how: with a flood of packets that consume all the bandwidth available for the connectivity, in proximity of the server network
2. Crash/hang the hardware or software operations in charge of the networking (TCP/UDP, IP in the Operating System or the network interface):
   how: exploit a vulnerability in the software or hardware of networking layers
3. Crash/hang the application running on the server and its backend (e.g., a database):
   how: exploit a vulnerability or some mechanisms to hang the server application





From this definition, you can see that there are several categories of resources that could be attacked:

• Network bandwidth

• System resources

• Application resources

Network bandwidth relates to the capacity of the network links connecting a server to the wider Internet. For most organizations, this is their connection to their Internet service provider (ISP), as shown in the example network in Figure 7.1. Usually this connection will have a lower capacity than the links within and between ISP routers. This means it is possible for more traffic to arrive at the ISP's routers over these higher-capacity links than can be carried over the link to the organization. In this circumstance, the router must discard some packets, delivering only as many as can be handled by the link. In normal network operation

such high loads might occur to a popular server experiencing traffic from a large number of legitimate users. A random portion of these users will experience a degraded or nonexistent service as a consequence. This is expected behavior for an overloaded TCP/IP network link. In a DoS attack, the vast majority of traffic directed at the target server is malicious, generated either directly or indirectly by the attacker. This traffic overwhelms any legitimate traffic, effectively denying legitimate users access to the server. Some recent high volume attacks have even been directed at the ISP network supporting the target organization, aiming to disrupt its connections to other networks. A number of recent DDoS attacks are listed in [AROR11], with comments on their growth in volume and impact.

A DoS attack targeting system resources typically aims to overload or crash its network handling software. Rather than consuming bandwidth with large volumes of traffic, specific types of packets are sent that consume the limited resources available on the system. These include temporary buffers used to hold arriving packets, tables of open connections, and similar memory data structures. The **SYN spoofing** attack, which we discuss next, is of this type. It targets the table of TCP connections on the server.

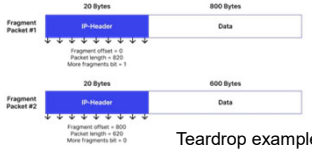
Another form of system resource attack uses packets whose structure triggers a bug in the system's network handling software, causing it to crash. This means the system can no longer communicate over the network until this software is reloaded, generally by rebooting the target system. This is known as a **poison packet** . The classic *ping of death* and *teardrop* attacks, directed at older Windows 9x systems, were of this form. These targeted bugs in the Windows network code that handled ICMP (Internet Control Message Protocol) echo request packets and packet fragmentation, respectively.

An attack on a specific application, such as a Web server, typically involves a number of valid requests, each of which consumes significant resources. This then limits the ability of the server to respond to requests from other users. For example, a Web server might include the ability to make database queries. If a large, costly query can be constructed, then an attacker could generate a large number of these that severely load the server. This limits its ability to respond to valid requests from other users.
DoS attacks may also be characterized by how many systems are used to direct traffic at the target system. Originally only one, or a small number of source systems directly under the attacker's control, was used. This is all that is required to send the packets needed for any attack targeting a bug in a server's network handling code or

some application. Attacks requiring high traffic volumes are more commonly sent from multiple systems at the same time, using distributed or amplified forms of DoS attacks. We discuss these later in this chapter.

IP Packet attacks (not network flooding) – System resource exhaustion or crash

- **IP malformation attack:**
  - Random optional fields of IP packet:
    E.g., all qualities of service bits are set to 1, victims apply some additional time to analyze the packet and slow processing.
    Flood of packets will hang the OS.
- **IP fragmentation attack** (notably, **Teardrop**)
  - Example: Fragments have overlapping sequences
  - There are many others:
    https://en.wikipedia.org/wiki/IP_fragmentation_attack
- **Land attack**
  - TCP/IP version: Src IP + Src Port == Dst IP + Dst Port
  - Requires *spoofing* of source IP address: fake source address is declared
  - Spoofed TCP SYN packet (connection initiation) with the target host's IP address to an open port as both source and destination. This causes the machine to reply to itself continuously till exhausting resources.
- **Targa3**
  - Uncommon IP packets consist of invalid fragmentation, protocol, packet size, header values, options, offsets, TCP segments, and routing flag. It caused crash of Windows.



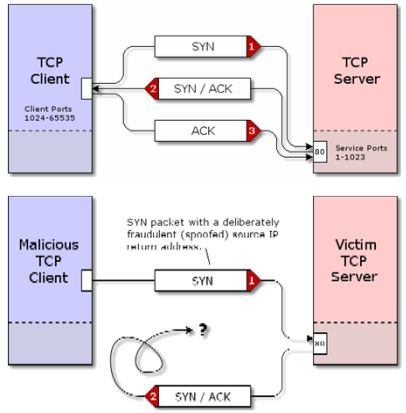
Teardrop example

ICMP – single packet attack
Ping of Death

- **Ping of Death**:
  - ICMP fragment size larger than the max size (65507, some bytes are used for ICMP header)
  - Crash of old versions of Unix, Linux, Mac, Windows, etc.
  - Solution:
    - Firewalls might drop malformed packets
    - Limit ICMP packet traffic only from some sources or IP ranges

SYN spoofing (TCP/IP) – 1/2 – (not network flooding)

- Common DoS attack
- It is NOT BASED on exhausting of network bandwidth, but the resources on a server machine
- Attacks the ability of a server to respond to future connection requests by overflowing the tables used to manage them
- Thus, legitimate users are denied access to the server
- Hence an attack on system resources, specifically the network handling code in the operating system



https://library.mobrien.com/Manuals/MPRM_Group/tcp_connections_101.htm

Along with the basic flooding attack, the other common classic DoS attack is the SYN spoofing attack. This attacks the ability of a network server to respond to TCP connection requests by overflowing the tables used to manage such connections. This means future connection requests from legitimate users fail, denying them access to the server. It is thus an attack on system resources, specifically the network handling code in the operating system.

SYN spoofing (TCP/IP) – 2/2

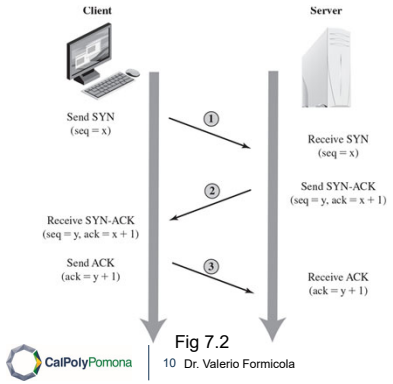


Fig 7.2

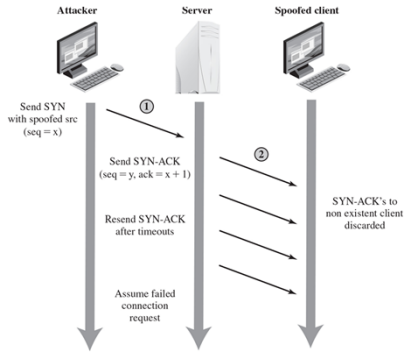


Fig 7.3

To understand the operation of these attacks, we need to review the three-way handshake that TCP uses to establish a connection. This is illustrated in Figure 7.2 . The client system initiates the request for a TCP connection by sending a SYN packet to the server. This identifies the client's address and port number and supplies an initial sequence number. It may also include a request for other TCP options. The server records all the details about this request in a table of known TCP connections. It then responds to the client with a SYN-ACK packet. This includes a sequence number for the server and increments the client's sequence number to confirm receipt of the SYN packet. Once the client receives this, it sends an ACK packet to the server with an incremented server sequence number and marks the connection as established. Likewise, when the server receives this ACK packet, it also marks the connection as established. Either party may then proceed with data transfer. In practice, this ideal exchange sometimes fails. These packets are transported using IP, which is an unreliable, though best-effort, network protocol. Any of the packets might be lost in transit, as a result of congestion, for example. Hence both the client and server keep track of which packets they have sent and, if no response is received in a reasonable time, will resend those packets. As a result, TCP is a reliable transport protocol, and any applications using it need not concern themselves with problems of lost or reordered packets. This does, however, impose an overhead on the systems in managing this reliable transfer of

packets.The sequence of events are as follows. 1. Client sends S Y N, s e q = x, to the server. The server receives the S Y N sequence, where s e q = x. 2. The server sends the S Y N dash A C K, s e q = y, a c k = x + 1, to the client. The client receives the acknowledgement. 3. The client sends the acknowledgement, a c k = x + 1, to the server. The server receives the acknowledgement.

A SYN spoofing attack exploits this behavior on the targeted server system. The attacker generates a number of SYN connection request packets with forged source addresses. For each of these the server records the details of the TCP connection request and sends the SYN-ACK packet to the claimed source address, as shown in Figure 7.3 . If there is a valid system at this address, it will respond with a RST (reset) packet to cancel this unknown connection request. When the server receives this packet, it cancels the connection request and removes the saved information. However, if the source system is too busy, or there is no system at the forged address, then no reply will return. In these cases the server will resend the SYN-ACK packet a number of times before finally assuming the connection request has failed and deleting the information saved concerning it. In this period between when the original SYN packet is received and when the server assumes the request has failed, the server is using an entry in its table of known TCP connections. This table is typically sized on the assumption that most connection requests quickly succeed and that a reasonable number of requests may be handled simultaneously. However, in a SYN spoofing attack, the attacker directs a very large number of forged connection requests at the targeted server. These rapidly fill the table of known TCP connections on the server. Once this table is full, any future requests, including legitimate requests from other users, are rejected. The table entries will time out and be removed, which in normal network usage corrects temporary overflow problems. However, if the attacker keeps a sufficient volume of forged requests flowing, this table will be constantly full and the server will be effectively cut off from the Internet, unable to respond to most legitimate connection requests.

In order to increase the usage of the known TCP connections table, the attacker ideally wishes to use addresses that will not respond to the SYN-ACK with a RST. This can be done by overloading the host that owns the chosen spoofed source address, or by simply using a wide range of random addresses. In this case, the attacker relies on the fact that there are many unused addresses on the Internet. Consequently, a reasonable proportion of randomly generated addresses will not correspond to a real host.

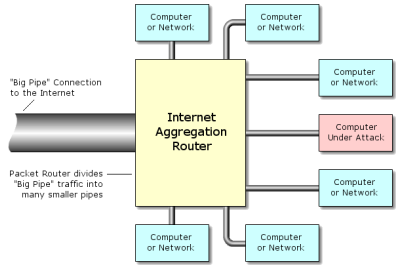
There is a significant difference in the volume of network traffic between a SYN spoof attack and the basic flooding attack we

discussed. The actual volume of SYN traffic can be comparatively low, nowhere near the maximum capacity of the link to the server. It simply has to be high enough to keep the known TCP connections table filled. Unlike the flooding attack, this means the attacker does not need access to a high-volume network connection. In the network shown in Figure 7.1 , the medium-sized organization, or even a broadband home user, could successfully attack the large company server using a SYN spoofing attack.

A flood of packets from a single server or a SYN spoofing attack originating on a single system were probably the two most common early forms of DoS attacks. In the case of a flooding attack this was a significant limitation, and attacks evolved to use multiple systems to increase their effectiveness. We next examine in more detail some of the variants of a flooding attack. These can be launched either from a single or multiple systems, using a range of mechanisms, which we explore.The events are as follows. 1. An attacker sends the S Y N with spoofed s r c, s e q = x, to the server. 2. The server sends the the spoofed S Y N acknowledgement to the spoofed client. S Y N acknowledgement to the non existent client discarded. Send S Y N acknowledgement sequence = y, a c k = x + 1, to the client. Resend S Y N acknowledgement after timeouts to the client. Assume failed connection request.

Flooding attacks: ICMP, SYN and UDP flooding

- Can use any kind of protocol to generate a high volume of traffic towards the victim IP address
- They are all based on the principle of congesting the destination network link



Flooding attacks take a variety of forms, based on which network protocol is being used to implement the attack. In all cases the intent is generally to overload the network capacity on some link to a server. The attack may alternatively aim to overload the server's ability to handle and respond to this traffic. These attacks flood the network link to the server with a torrent of malicious packets competing with, and usually overwhelming, valid traffic flowing to the server. In response to the congestion this causes in some routers on the path to the targeted server, many packets will be dropped. Valid traffic has a low probability of surviving discard caused by this flood and hence of accessing the server. This results in the server's ability to respond to network connection requests being either severely degraded or failing entirely.

Virtually any type of network packet can be used in a flooding attack. It simply needs to be of a type that is permitted to flow over the links toward the targeted system, so that it can consume all available capacity on some link to the target server. Indeed, the larger the packet, the more effective is the attack. Common flooding attacks use any of the ICMP, UDP, or TCP SYN packet types. It is even possible to flood with some other IP packet type. However, as these are less common and their usage more targeted, it is easier to filter

for them and hence hinder or block such attacks.

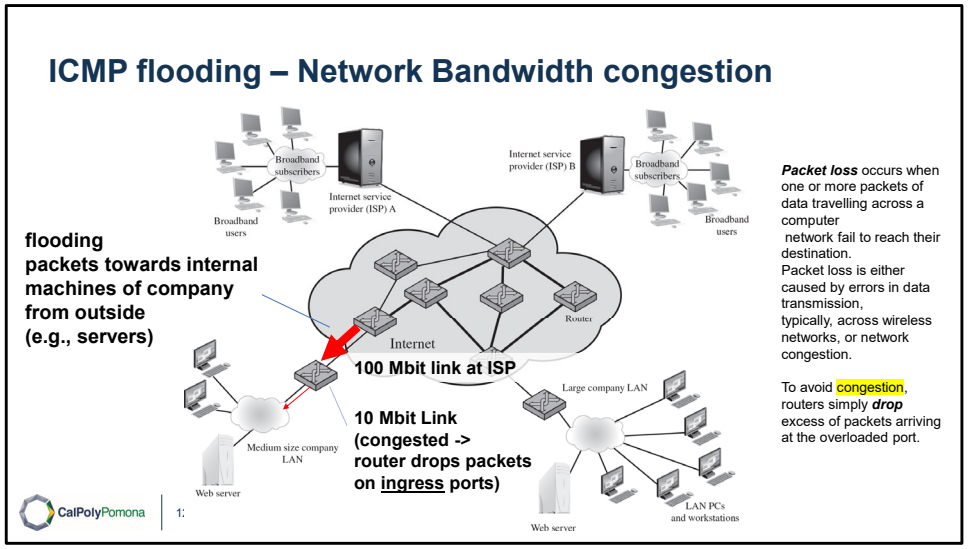
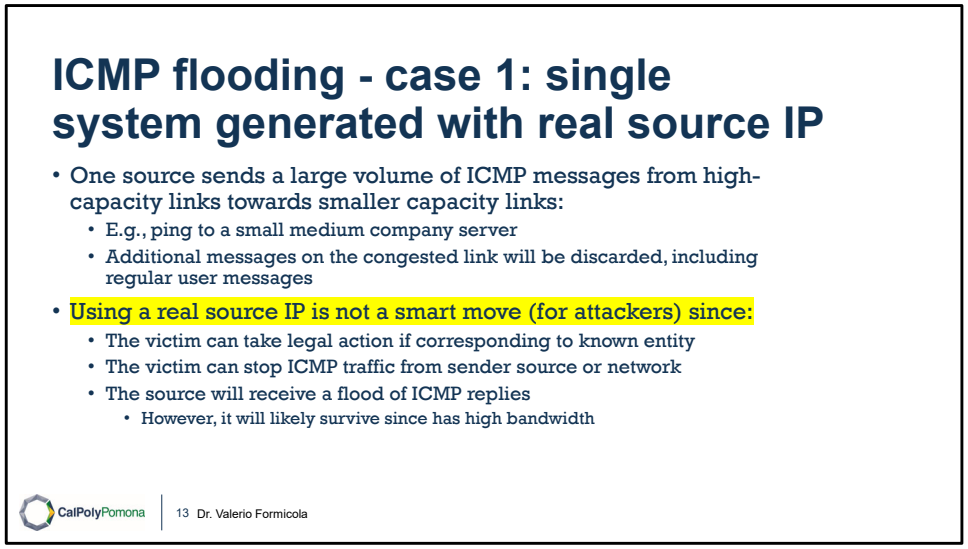


Figure 7.1 Example Network to Illustrate DoS Attacks

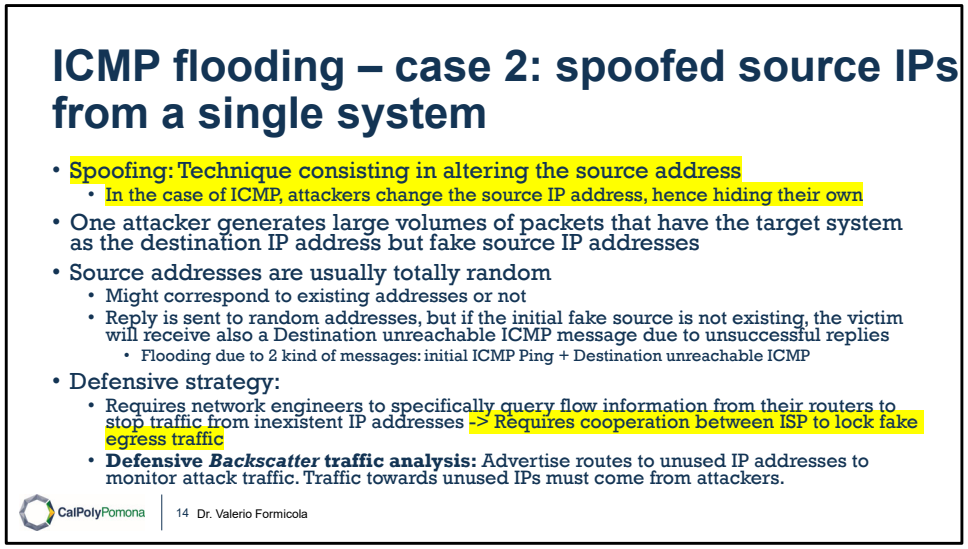
The simplest classical DoS attack is a flooding attack on an organization. The aim of this attack is to overwhelm the capacity of the network connection to the target organization. If the attacker has access to a system with a higher-capacity network connection, then this system can likely generate a higher volume of traffic than the lower-capacity target connection can handle. For example, in the network shown in Figure 7.1 , the attacker might use the large company's Web server to target the medium-sized company with a lower-capacity network connection. The attack might be as simple as using a flooding ping command directed at the Web server in the target company. This traffic can be handled by the higher-capacity links on the path between them, until the final router in the Internet cloud is reached. At this point some packets must be discarded, with the remainder consuming most of the capacity on the link to the medium-sized company. Other valid traffic will have little chance of surviving discard as the router responds to the resulting congestion on this link.

In this classic ping flood attack, the source of the attack is clearly identified since its address is used as the source address in the ICMP echo request packets. This has two disadvantages from the attacker's perspective. First, the source of the attack is explicitly identified, increasing the chance that the attacker can be identified and legal action taken in response. Second, the targeted system will attempt to respond to the packets being sent. In the case of any ICMP echo request packets received by the server, it would respond to each with an ICMP echo response packet directed back to the sender. This effectively reflects the attack back at the source system. Since the source system has a higher network bandwidth, it is more likely to survive this reflected attack. However, its network performance will be noticeably affected, again increasing the chances of the attack being detected and action taken in response. For both of these reasons the attacker would like to hide the identity of the source system. This means that any such attack packets need to use a falsified, or spoofed, address.



The ping flood using ICMP echo request packets is a classic example of an ICMP flooding attack. This type of ICMP packet was chosen since traditionally network administrators allowed such packets into their networks, as ping is a useful network diagnostic tool. More recently, many organizations have restricted the ability of these packets to pass through their firewalls. In response, attackers have started using other ICMP packet types. Since some of these should be handled to allow the correct operation of TCP/IP, they are much more likely to be allowed through an organization's firewall. Filtering some of these critical ICMP packet types would degrade or break normal TCP/IP network behavior. ICMP destination unreachable and time exceeded packets are examples of such critical packet types.

An attacker can generate large volumes of one of these packet types. Because these packets include part of some notional erroneous packet that supposedly caused the error being reported, they can be made comparatively large, increasing their effectiveness in flooding the link. ICMP flood attacks remain one of the most common types of DDoS attacks [SYMA16].



A common characteristic of packets used in many types of DoS attacks is the use of forged source addresses. This is known as source address spoofing. Given sufficiently privileged access to the network handling code on a computer system, it is easy to create packets with a forged source address (and indeed any other attribute that is desired). This type of access is usually via the raw socket interface on many operating systems. This interface was provided for custom network testing and research into network protocols. It is not needed for normal network operation. However, for reasons of historical compatibility and inertia, this interface has been maintained in many current operating systems. Having this standard interface available greatly eases the task of any attacker trying to generate packets with forged attributes. Otherwise an attacker would most likely need to install a custom device driver on the source system to obtain this level of access to the network, which is much more error prone and dependent on operating system version.

Given raw access to the network interface, the attacker now generates large volumes of packets. These would all have the target system as the destination address but would use randomly selected, usually different, source addresses for each packet. Consider the flooding ping example from the previous section. These custom ICMP echo

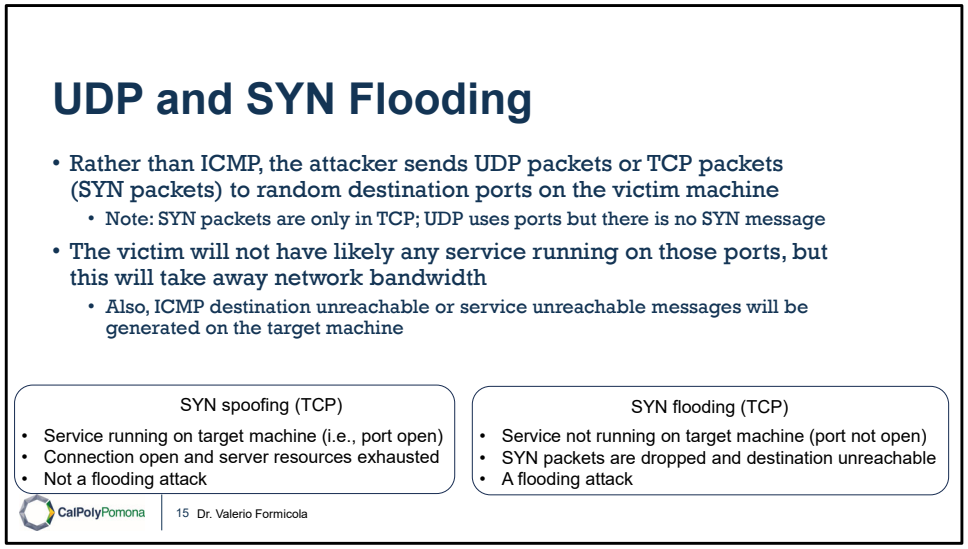
request packets would flow over the same path from the source toward the target system. The same congestion would result in the router connected to the final, lower capacity link. However, the ICMP echo response packets, generated in response to those packets reaching the target system, would no longer be reflected back to the source system. Rather they would be scattered across the Internet to all the various forged source addresses. Some of these addresses might correspond to real systems. These might respond with some form of error packet, since they were not expecting to see the response packet received. This only adds to the flood of traffic directed at the target system. Some of the addresses may not be used or may not reachable. For these, ICMP destination unreachable packets might be sent back. Or these packets might simply be discarded. Any response packets returned only add to the flood of traffic directed at the target system.

In addition, the use of packets with forged source addresses means the attacking system is much harder to identify. The attack packets seem to have originated at addresses scattered across the Internet. Hence, just inspecting each packet's header is not sufficient to identify its source. Rather the flow of packets of some specific form through the routers along the path from the source to the target system must be identified. This requires the cooperation of the network engineers managing all these routers and is a much harder task than simply reading off the source address. It is not a task that can be automatically requested by the packet recipients. Rather it usually requires the network engineers to specifically query flow information from their routers. This is a manual process that takes time and effort to organize.

It is worth considering why such easy forgery of source addresses is allowed on the Internet. It dates back to the development of TCP/IP, which occurred in a generally cooperative, trusting environment. TCP/IP simply does not include the ability, by default, to ensure that the source address in a packet really does correspond with that of the originating system. It is possible to impose filtering on routers to ensure this (or at least that source network address is valid). However, this filtering needs to be imposed as close to the originating system as possible, where the knowledge of valid source addresses is as accurate as possible. In general, this should occur at the point where an organization's network connects to the wider Internet, at the borders of the ISP's providing this connection. Despite this being a long-standing security recommendation to combat problems such as DoS attacks, for example (RFC 2827), many ISPs do not implement such filtering. As a consequence, attacks using spoofed-source packets continue to occur frequently.

There is a useful side effect of this scattering of response packets to some original flow of spoofed-source packets. Security researchers, such as those with the Honeynet Project, have taken blocks of unused IP addresses, advertised routes to them, and then collected details of any packets sent to these addresses. Since no real systems use these addresses, no legitimate

packets should be directed to them. Any packets received might simply be corrupted. It is much more likely, though, that they are the direct or indirect result of network attacks. The ICMP echo response packets generated in response to a ping flood using randomly spoofed source addresses is a good example. This is known as backscatter traffic . Monitoring the type of packets gives valuable information on the type and scale of attacks being used, as described by [MOOR06], for example. This information is being used to develop responses to the attacks being seen.



An alternative to using ICMP packets is to use UDP packets directed to some port number, and hence potential service, on the target system. A common choice was a packet directed at the diagnostic echo service, commonly enabled on many server systems by default. If the server had this service running, it would respond with a UDP packet back to the claimed source containing the original packet data contents. If the service is not running, then the packet is discarded, and possibly an ICMP destination unreachable packet is returned to the sender. By then the attack has already achieved its goal of occupying capacity on the link to the server. Just about any UDP port number can be used for this end. Any packets generated in response only serve to increase the load on the server and its network links.

Spoofed source addresses are normally used if the attack is generated using a single source system, for the same reasons as with ICMP attacks. If multiple systems are used for the attack, often the real addresses of the compromised, zombie, systems are used. When multiple systems are used, the consequences of both the reflected flow of packets and the ability to identify the attacker are reduced.

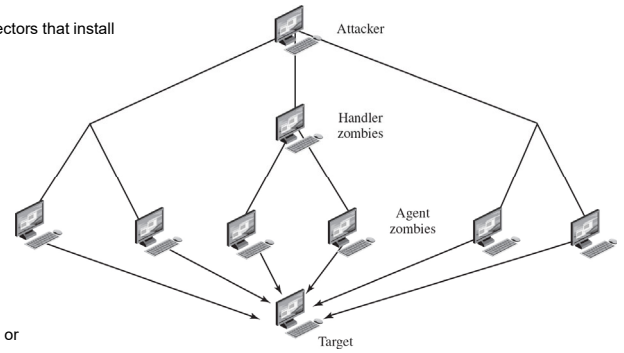
Another alternative is to send TCP packets to the target system. Most likely these would be normal TCP connection requests, with either real or spoofed source addresses. They would have an effect similar to the SYN spoofing attack we've described. In this case, though, it is the total volume of packets that is the aim of the attack rather than the system code. This is the difference between a SYN spoofing attack and a **SYN flooding** attack.

This attack could also use TCP data packets, which would be rejected by the server as not belonging to any known connection. But again, by this time the attack has already succeeded in flooding the links to the server.

All of these flooding attack variants are limited in the total volume of traffic that can be generated if just a single system is used to launch the attack. The use of a single system also means the attacker is easier to trace. For these reasons, a variety of more sophisticated attacks, involving multiple attacking systems, have been developed. By using multiple systems, the attacker can significantly scale up the volume of traffic that can be generated. Each of these systems need not be particularly powerful or on a high-capacity link. But what they don't have individually, they more than compensate for in large numbers. Also, by directing the attack through intermediaries, the attacker is further distanced from the target and significantly harder to locate and identify. Indirect attack types that utilize multiple systems include

Distributed Denial of Service – DDoS

- Use of multiple systems to generate attacks, previously infected somehow
  - Trojans, worms, viruses can be vectors that install or download the zombie agents
- A Botnet is a common strategy to implement the attack
- Each handler controls a subset of zombies
- The attacker simply connects to the handler to control the zombies.
- Handler to zombies connections might be encrypted
- The zombies then execute the flooding or spoofing attack



General model of a Botnet, some components might be combined

Recognizing the limitations of flooding attacks generated by a single system, one of the earlier significant developments in DoS attack tools was the use of multiple systems to generate attacks. These systems were typically compromised user workstations or PCs. The attacker uses malware to subvert the system and to install an attack agent, which they can control. Such systems are known as zombies. Large collections of such systems under the control of one attacker can be created, collectively forming a botnet , as we discussed in Chapter 6. Such networks of compromised systems are a favorite tool of attackers, and can be used for a variety of purposes, including distributed denial-of-service (DDoS) attacks. Indeed, there is an underground economy that creates and hires out botnets for use in such attacks. [SYMA16] report evidence that 40% of DDoS attacks in 2015 were from such botnets for hire. In the example network shown in Figure 7.1, some of the broadband user systems may be compromised and used as zombies to attack any of the company or other links shown.

Many other DDoS tools have been developed since. Instead of using dedicated handler programs, many now use an IRC or similar instant messaging server program to manage communications with the agents. Many of these more recent tools also use cryptographic mechanisms to authenticate the agents to the handlers, in order to hinder

analysis of command traffic.

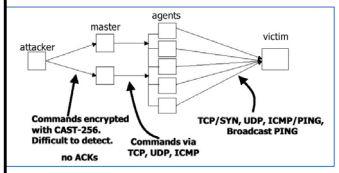
The best defense against being an unwitting participant in a DDoS attack is to prevent your systems from being compromised. This requires good system security practices and keeping the operating systems and applications on such systems current and patched.

For the target of a DDoS attack, the response is the same as for any flooding attack, but with greater volume and complexity. We discuss appropriate defenses and responses in Sections 7.6 and 7.7.

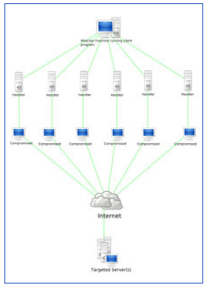
While the attacker could command each zombie individually, more generally a control hierarchy is used. A small number of systems act as handlers controlling a much larger number of agent systems, as shown in Figure 7.4 . There are a number of advantages to this arrangement. The attacker can send a single command to a handler, which then automatically forwards it to all the agents under its control. Automated infection tools can also be used to scan for and compromise suitable zombie systems, as we discuss in Chapter 6 . Once the agent software is uploaded to a newly compromised system, it can contact one or more handlers to automatically notify them of its availability. By this means, the attacker can automatically grow suitable botnets.

Examples of DDoS botnets – old tools

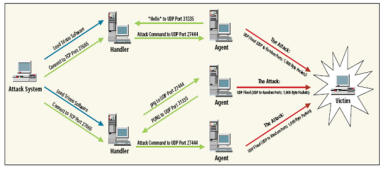
Tribe Flood Network 2000 - TFN2k



Stacheldraht



Trin00



The Trinoo architecture

Stacheldraht combines the features of Trin00 and TFN and adds encrypted communication between the client and the handler, plus automated remote update of the agents.
Zombies attacks:
Ping flood, UDP flood, TCP SYN flood, and Smurf attack

1. Attack the master which will host tools
2. Scan vulnerable machines to be elected as handlers
3. Attack zombies from handlers to install daemons
4. Execute commands from handlers to zombies
5. Zombies perform *UDP flood*

More here: https://digital.library.unt.edu/ark:/67531/metadc739628/m2/1/high_res_d/792253.pdf

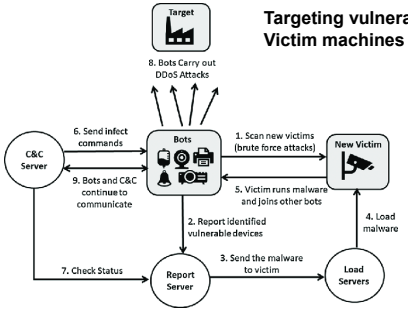
One of the earliest and best-known DDoS tools is Tribe Flood Network (TFN), written by the hacker known as Mixter. The original variant from the 1990s exploited Sun Solaris systems. It was later rewritten as Tribe Flood Network 2000 (TFN2K) and could run on UNIX, Solaris, and Windows NT systems. TFN and TFN2K use a version of the two-layer command hierarchy shown in Figure 7.4. The agent was a Trojan program that was copied to and run on compromised, zombie systems. It was capable of implementing ICMP flood, SYN flood, UDP flood, and ICMP amplification forms (later) of DoS attacks. TFN did not spoof source addresses in the attack packets. Rather it relied on a large number of compromised systems, and the layered command structure, to obscure the path back to the attacker. The agent also implemented some other rootkit functions as we describe in Chapter 6. The handler was simply a command-line program run on some compromised systems. The attacker accessed these systems using any suitable mechanism giving shell access, and then ran the handler program with the desired options. Each handler could control a large number of agent systems, identified using a supplied list. Communications between the handler and

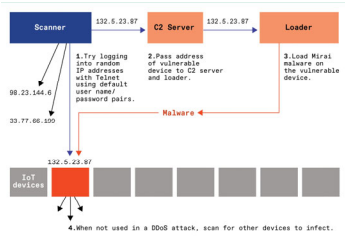
its agents was encrypted and could be intermixed with a number of decoy packets. This hindered attempts to monitor and analyze the control traffic. Both these communications and the attacks themselves could be sent via randomized TCP, UDP, and ICMP packets. This tool demonstrates the typical capabilities of a DDoS attack system.

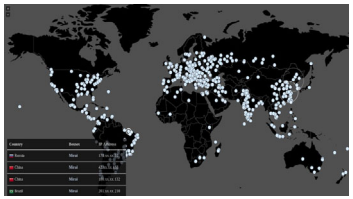
Many other DDoS tools have been developed since. Instead of using dedicated handler programs, many now use an IRC or similar instant messaging server program, or web-based HTTP servers, to manage communications with the agents. Many of these more recent tools also use cryptographic mechanisms to authenticate the agents to the handlers, in order to hinder analysis of command traffic.

Example recent DDoS – Mirai (2016)

Targeting vulnerable IoT.
Victim machines were DNS servers





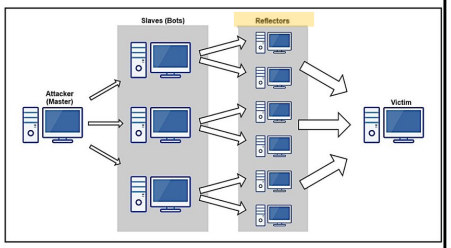


https://www.youtube.com/watch?v=cQkQpbkhzIo&t=9s\
Infection progression

DoS/DDoS - Reflection Attacks

- Attacker sends packets to a known service on the intermediary with a spoofed source address of the actual target system
- When intermediary responds, the response is sent to the target
- "Reflects" the attack off the intermediary (reflector)
  - Reflectors are NOT compromised hosts that simply reply to (spoofed) sender requests
- Goal is to generate enough volumes of packets to flood the link to the target system without alerting the intermediary
  - Very few reflectors are chosen, but with high-capacity bandwidth
- Lack of backscatter traffic: Very hard to detect based on the source
  - The basic defense against these attacks is blocking spoofed-source packets

Spoofed source IP == victim IP



For this attack, main categories of protocols are used to generate replies:
- ICMP messages, like ping
- UDP based protocols, e.g., DSN, SNMP, ISAKMP, NTP, UDP echo
- TCP SYN

In contrast to DDoS attacks, where the intermediaries are compromised systems running the attacker's programs, reflector and amplifier attacks use network systems functioning normally. The attacker sends a network packet with a spoofed source address to a service running on some network server. The server responds to this packet, sending it to the spoofed source address that belongs to the actual attack target. If the attacker sends a number of requests to a number of servers, all with the same spoofed source address, the resulting flood of responses can overwhelm the target's network link. The fact that normal server systems are being used as intermediaries, and that their handling of the packets is entirely conventional, means these attacks can be easier to deploy and harder to trace back to the actual attacker. There are two basic variants of this type of attack: the simple reflection attack and the amplification attack.

The **reflection attack** is a direct implementation of this type of attack. The attacker sends packets to a known service on the intermediary with a spoofed source address of the actual target system. When the intermediary responds, the response is sent to the target. Effectively this reflects the attack off the intermediary, which is termed the reflector, and is why this is called a reflection attack.

Ideally the attacker would like to use a service that created a larger response packet than the original request. This allows the attacker to convert a lower volume stream of packets from the originating system into a higher volume of packet data from the intermediary directed at the target. Common UDP services are often used for this purpose. Originally the echo service was a favored choice, although it does not create a larger response packet. However, any generally accessible UDP service could be used for this type of attack. The chargen, DNS, SNMP, or ISAKMP services have all been exploited in this manner, in part because they can be made to generate larger response packets directed at the target.

The intermediary systems are often chosen to be high-capacity network servers or routers with very good network connections. This means they can generate high volumes of traffic if necessary, and if not, the attack traffic can be obscured in the normal high volumes of traffic flowing through them. If the attacker spreads the attack over a number of intermediaries in a cyclic manner, then the attack traffic flow may well not be easily distinguished from the other traffic flowing from the system. This, combined with the use of spoofed source addresses, greatly increases the difficulty of any attempt to trace the packet flows back to the attacker's system.

Another variant of reflection attack uses TCP SYN packets and exploits the normal three-way handshake used to establish a TCP connection. The attacker sends a number of SYN packets with spoofed source addresses to the chosen intermediaries. In turn the intermediaries respond with a SYN-ACK packet to the spoofed source address, which is actually the target system. The attacker uses this attack with a number of intermediaries. The aim is to generate high enough volumes of packets to flood the link to the target system. The target system will respond with a RST packet for any that get through, but by then the attack has already succeeded in overwhelming the target's network link.

This attack variant is a flooding attack that differs from the SYN spoofing attack we discussed earlier in this chapter. The goal is to flood the network link to the target, not to exhaust its network handling resources. Indeed, the attacker would usually take care to limit the volume of traffic to any particular intermediary to ensure that it is not overwhelmed by, or even notices, this traffic. This is both because its continued correct functioning is an essential component of this attack, as is limiting the chance of the attacker's actions being detected. The 2002 attack on GRC.com was of this form. It used connection requests to the BGP routing service on core routers as the

primary intermediaries. These generated sufficient response traffic to completely block normal access to GRC.com. However, as GRC.com discovered, once this traffic was blocked, a range of other services, on other intermediaries, were also being used. GRC noted in its report on this attack that “you know you’re in trouble when packet floods are competing to flood you.”

Any generally accessible TCP service can be used in this type of attack. Given the large number of servers available on the Internet, including many well-known servers with very high capacity network links, there are many possible intermediaries that can be used. What makes this attack even more effective is that the individual TCP connection requests are indistinguishable from normal connection requests directed to the server. It is only if they are running some form of intrusion detection system that detects the large numbers of failed connection requests from one system that this attack might be detected and possibly blocked. If the attacker is using a number of intermediaries, then it is very likely that even if some detect and block the attack, many others will not, and the attack will still succeed.

UDP Echo protocol

- Similar to ICMP Echo Request and Reply, but in TCP and UDP
- UDP Echo measures round-trip delay.
- UDP Echo tests return Latency and Packet Loss results.
- UDP tests must be configured to target a destination port that is listening for UDP traffic. For example, many Cisco devices running IOS IP SLAs use port 1967 and UNIX systems use port 7. Failure to configure the port can result in timeouts for UDP tests. Set the Destination Port parameter in the General options for UDP tests.
- Existing also in TCP

```
nc -uvn <IP> 7
Hello echo    #This is what you send
Hello echo    #This is the response
```

Example: DNS Reflection Attack with UDP Echo

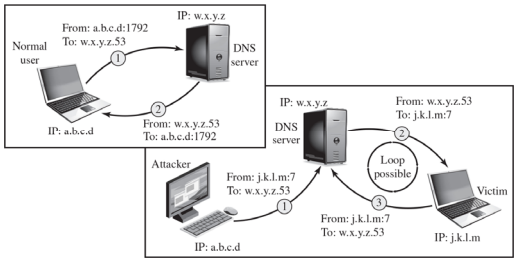


Figure 7.6

Attack Model:

- Spoofed DNS request (over UDP protocol) from attacker towards a DSN server (reflector):
  Source IP = Victim IP
  Source Port = 7
- DNS server will reply to Victim IP and send the UDP message to Port 7 of Victim
- If Victim has UDP service activated, then victim will reply to the UDP Echo, by sending another UDP Echo to DNS server 7
- This activates an infinite loop
- Can be stopped:
  - Disable UDP echo from external networks on any machines
  - Detect DNS (UDP) requests with 7 as a source port

A further variation of the reflector attack establishes a self-contained loop between the intermediary and the target system. Both systems act as reflectors. Figure 7.6 shows this type of attack. The upper part of the figure shows normal Domain Name System operation. The DNS client sends a query from its UDP port 1792 to the server's DNS port 53 to obtain the IP address of a domain name. The DNS server sends a UDP response packet including the IP address. The lower part of the figure shows a reflection attack using DNS. The attacker sends a query to the DNS server with a spoofed IP source address of j.k.l.m; this is the IP address of the target. The attacker uses port 7, which is usually associated with echo, a reflector service. The DNS server then sends a response to the victim of the attack, j.k.l.m, addressed to port 7. If the victim is offering the echo service, it may create a packet that echoes the received data back to the DNS server. This can cause a loop between the DNS server and the victim if the DNS server responds to the packets sent by the victim. Most reflector attacks can be prevented through network-based and host-based firewall rulesets that reject suspicious combinations of source and destination ports.

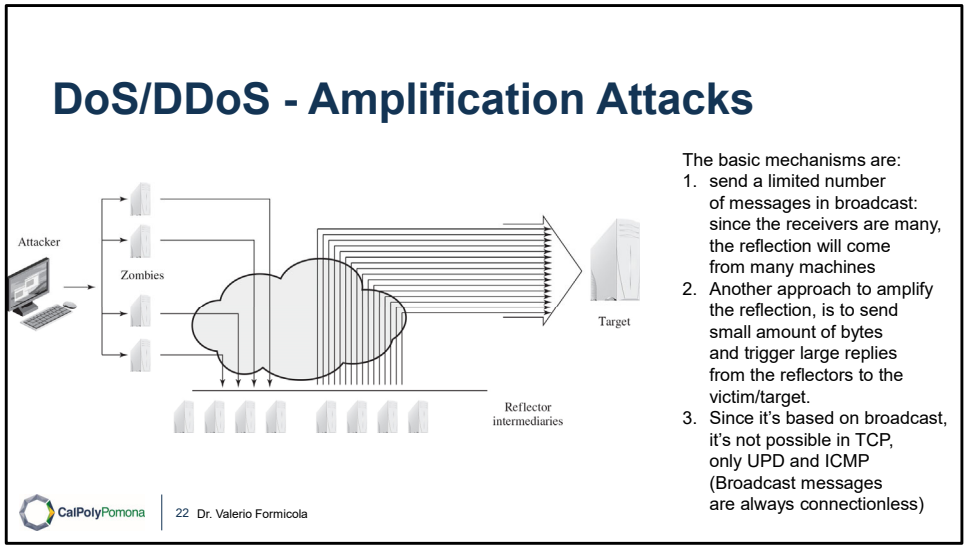
While very effective if possible, this type of attack is fairly easy to filter for because the combinations of

service ports used should never occur in normal network operation.

When implementing any of these reflection attacks, the attacker could use just one system as the original source of packets. This suffices, particularly if a service is used that generates larger response packets than those originally sent to the intermediary. Alternatively, multiple systems might be used to generate higher volumes of traffic to be reflected and to further obscure the path back to the attacker. Typically a botnet would be used in this case.

Another characteristic of reflection attacks is the lack of backscatter traffic. In both direct flooding attacks and SYN spoofing attacks, the use of spoofed source addresses results in response packets being scattered across the Internet and thus detectable. This allows security researchers to estimate the volumes of such attacks. In reflection attacks, the spoofed source address directs all the packets at the desired target and any responses to the intermediary. There is no generally visible side effect of these attacks, making them much harder to quantify. Evidence of them is only available from either the targeted systems and their ISPs or the intermediary systems. In either case, specific instrumentation and monitoring would be needed to collect this evidence.

Fundamental to the success of reflection attacks is the ability to create spoofed-source packets. If filters are in place that block spoofed-source packets, as described in (RFC 2827), then these attacks are simply not possible. This is the most basic, fundamental defense against such attacks. This is not the case with either SYN spoofing or flooding attacks (distributed or not). They can succeed using real source addresses, with the consequences already noted.



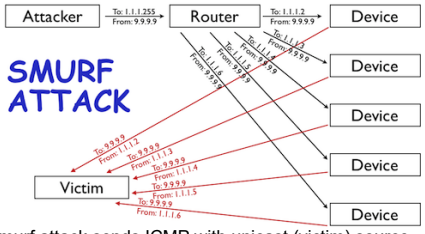
Amplification attacks are a variant of reflector attacks and also involve sending a packet with a spoofed source address for the target system to intermediaries. They differ in generating multiple response packets for each original packet sent. This can be achieved by directing the original request to the broadcast address for some network. As a result, all hosts on that network can potentially respond to the request, generating a flood of responses as shown in Figure 7.7 . It is only necessary to use a service handled by large numbers of hosts on the intermediate network. A ping flood using ICMP echo request packets was a common choice, since this service is a fundamental component of TCP/IP implementations and was often allowed into networks. The well-known *smurf* DoS program used this mechanism and was widely popular for some time. Another possibility is to use a suitable UDP service, such as the echo service. The *fraggle* program implemented this variant. Note that TCP services cannot be used in this type of attack; because they are connection oriented, they cannot be directed at a broadcast address. Broadcasts are inherently connectionless.

The best additional defense against this form of attack is to not allow directed broadcasts to be routed into a

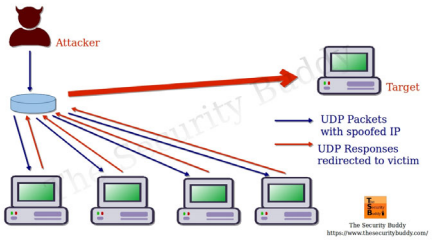
network from outside. Indeed, this is another longstanding security recommendation, unfortunately about as widely implemented as that for blocking spoofed source addresses. If these forms of filtering are in place, these attacks cannot succeed. Another defense is to limit network services like echo and ping from being accessed from outside an organization. This restricts which services could be used in these attacks, at a cost in ease of analyzing some legitimate network problems.

Attackers scan the Internet looking for well-connected networks that do allow directed broadcasts and that implement suitable services attackers can reflect off. These lists are traded and used to implement such attacks.

Example of DDoS Amplification: Smurf and Fraggle attacks



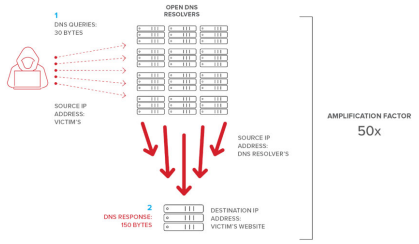
- Smurf attack sends ICMP with unicast (victim) source and broadcast as a destination to reach the reflectors
- Can also be the other way around: broadcast source, unicast reflectors



Fraggle is like Smurf but using UDP Echo as a message request and reply (still IP broadcast used as a destination)

Example of DDoS Amplification: DNS Amplification Attacks

- Use packets directed at a legitimate DNS server as the intermediary system
- Attacker creates a series of DNS requests containing the spoofed source address of the target system
- Exploit DNS behavior to convert a small request to a much larger response (amplification)
- Target is flooded with responses
- Basic defense against this attack is to prevent the use of spoofed source addresses



https://cyberhoot.com/cybrary/dns-reflection-attack/

In addition to the DNS reflection attack discussed previously, a further variant of an amplification attack uses packets directed at a legitimate DNS server as the intermediary system. Attackers gain attack amplification by exploiting the behavior of the DNS protocol to convert a small request into a much larger response. This contrasts with the original amplifier attacks, which use responses from multiple systems to a single request to gain amplification. Using the classic DNS protocol, a 60-byte UDP request packet can easily result in a 512-byte UDP response, the maximum traditionally allowed. All that is needed is a name server with DNS records large enough for this to occur.

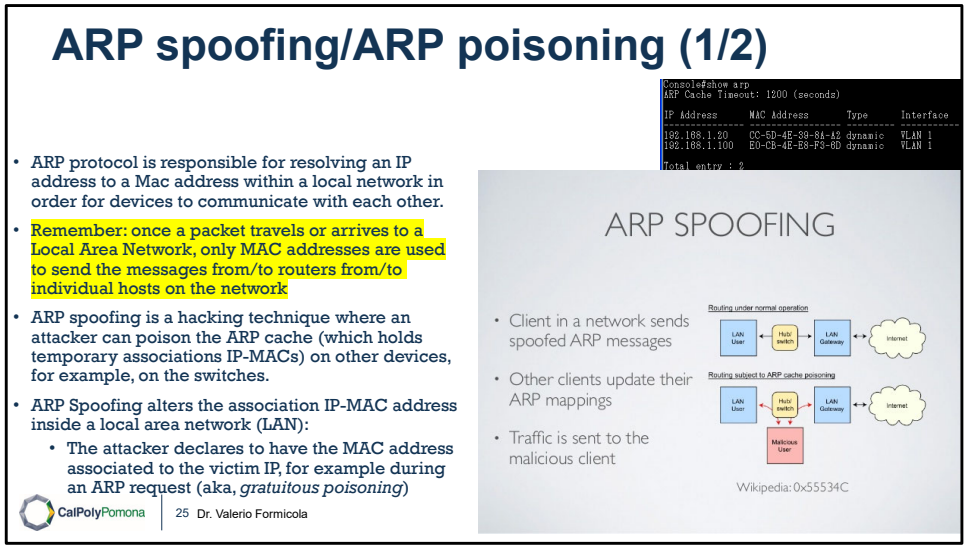
These attacks have been seen for several years. More recently, the DNS protocol has been extended to allow much larger responses of over 4000 bytes to support extended DNS features such as IPv6, security, and others. By targeting servers that support the extended DNS protocol, significantly greater amplification can be achieved than with the classic DNS protocol.

In this attack, a selection of suitable DNS servers with good network connections are chosen. The attacker

creates a series of DNS requests containing the spoofed source address of the target system. These are directed at a number of the selected name servers. The servers respond to these requests, sending the replies to the spoofed source, which appears to them to be the legitimate requesting system. The target is then flooded with their responses. Because of the amplification achieved, the attacker need only generate a moderate flow of packets to cause a larger, amplified flow to flood and overflow the link to the target system. Intermediate systems will also experience significant loads. By using a number of high-capacity, well-connected systems, the attacker can ensure that intermediate systems are not overloaded, allowing the attack to proceed.

A further variant of this attack exploits recursive DNS name servers. This is a basic feature of the DNS protocol that permits a DNS name server to query a number of other servers to resolve a query for its clients. The intention was that this feature is used to support local clients only. However, many DNS systems support recursion by default for any requests. They are known as open recursive DNS servers. Attackers may exploit such servers for a number of DNS-based attacks, including the DNS amplification DoS attack. In this variant, the attacker targets a number of open recursive DNS servers. The name information being used for the attack need not reside on these servers, but can be sourced from anywhere on the Internet. The results are directed at the desired target using spoofed source addresses.

Like all the reflection-based attacks, the basic defense against these is to prevent the use of spoofed source addresses. Appropriate configuration of DNS servers, in particular limiting recursive responses to internal client systems only, as described in RFC 5358, can restrict some variants of this attack.



An attacker device scans the network, then finds one device that he want to compromise, and sends fake packets (ARP packets) to that specific device (using that IP that is already known). Using the switch example, the switch is then updated using the ARP cache. After this has been completed, he can redirect any traffic coming from that switch and can then access that device to pass data from the compromised device to his own. When information is stolen in this way, the victim does not realize his data was captured.

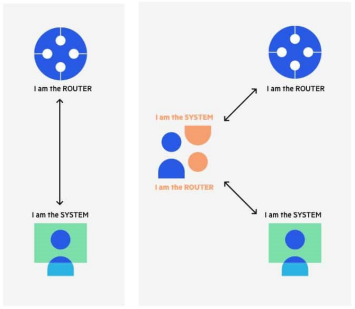
This is what's known as a MITM (Man in the middle) attack.

By sending frames to the compromised device, the hacker than make the compromised user believe that they have the correct Mac addresses however the frame is being sent straight to the attacker for analysis. The hacker can receive; forward or even manipulate that data. It is usually switches that are most vulnerable, especially those with default configuration.

ARP cache poisoning takes advantage of the nature of the ARP protocol, devices using ARP accept updates anytime, which means any device in the network can send the ARP reply packet to another device, updating its ARP cache to a new value.

ARP spoofing/ARP poisoning (2/2): Man-In-The-Middle (MITM) example

The ARP spoofing attacker pretends to be both sides of a network communication channel



The attacker can advertise ARP responses to have:
- the MAC address associated to the router IP
- the MAC address associated to the victim IP

If the attacker declares to be the router, it can intercept all the messages towards the router (usually packets that leave the LAN).

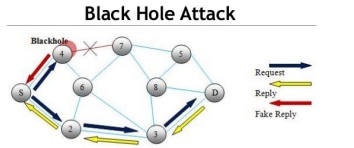
If the attacker declares also to have the MAC of an internal victim host, it can intercept all the messages for the victim.

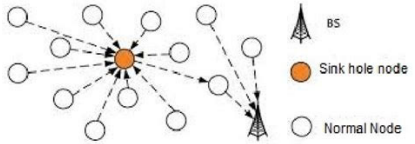
Perfect example of how to implement a **Man-In-The-Middle (MITM)** attack in a Local Area Network.

What the attacker can do now on:
1. Sniff the packets and steal cleartext data
2. Perform session hijacking—if the attacker obtains a session ID, they can gain access to accounts the user is currently logged into.
3. Alter communication—for example pushing a malicious file or website to the workstation.
4. DDoS - an attacker might send out ARP Response messages that falsely map hundreds or even thousands of IP addresses to a single MAC address, potentially overwhelming the target machine.
5. Blackhole DoS attack – attacker might simply drop packets for the victim, hence interrupting services, especially if the victim might is a server.

DoS in mobile ad hoc networks – Blackhole and Sinkhole

- In wireless networks some protocols (e.g., Zigbee) build routing paths based on quality of wireless paths and hop counts (e.g., AODV routing protocol)
- By advertising fake routes or less hop counts, a malicious node will force an attacker to send all data to the malicious node that can discard all the packets

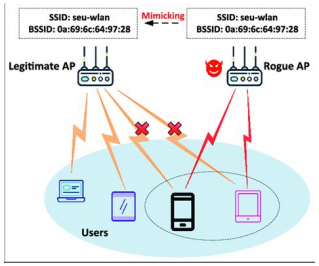


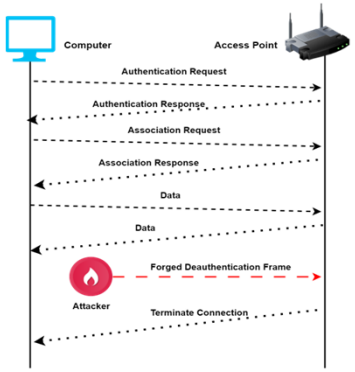


Normally, all paths reach the BS - Base station
During the attack, all traffic goes to the Sinkhole which can discard packets or perform Man in the Middle

DoS in WiFi – Rogue AP and Disassociation attacks

Rogue AP can act as the legitimate AP and disconnect or drop packets to the victims.
Note: BSSID is the wifi MAC of the AP

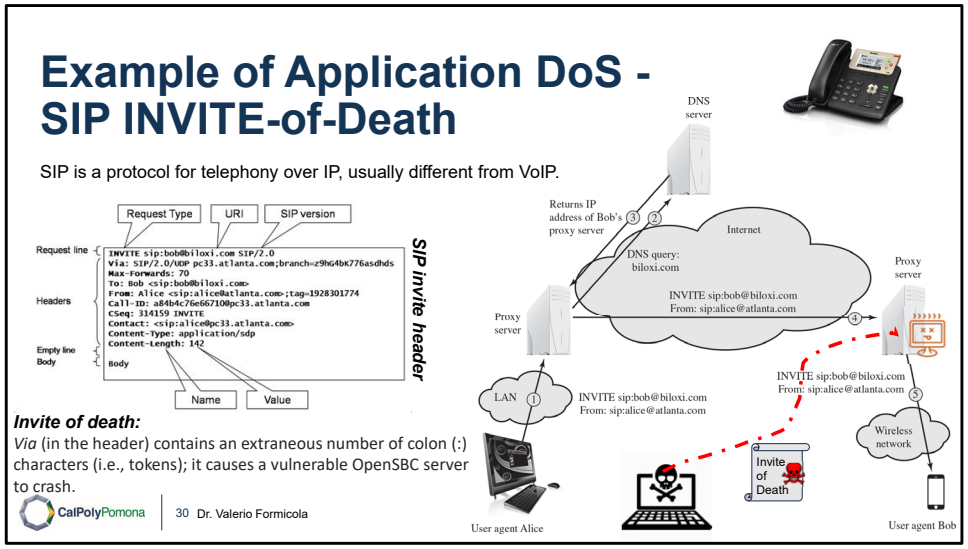




Disassociation attack performs a MAC spoofing of victim machine, and performs a request of disassociation to the AP, hence disconnecting the victim from the wireless network

Application-level DoS attacks

- In theory, DNS attacks are also Application-level attacks since DNS is an application using UDP (well-known port 53)
  - This attack aims at consuming network bandwidth toward the victim
- On the other hand, application-level DoS attacks are meant to exhaust some mechanisms in the application to generate unproductive operation of a server
- Examples:
  - Huge **amount of invitations** in voice and conferencing applications: SIP invite flood
  - **Heavy requests** from clients, which will end up being overloaded **in the backend servers:**
    - Cyberslam, Slowris, HTTP flood
  - Generate a **crash** in the backend server**:** SIP Invite of death, RegularExpressionDoS (ReDoS)



Voice over IP (VoIP) telephony is now widely deployed over the Internet. The standard protocol used for call setup in VoIP is the Session Initiation Protocol (SIP). SIP is a text-based protocol with a syntax similar to that of HTTP. There are two different types of SIP messages: requests and responses. Figure 7.5 is a simplified illustration of the operation of the SIP INVITE message, used to establish a media session between user agents. In this case, Alice's user agent runs on a computer, and Bob's user agent runs on a cell phone. Alice's user agent is configured to communicate with a proxy server (the outbound server) in its domain and begins by sending an INVITE SIP request to the proxy server that indicates its desire to invite Bob's user agent into a session. The proxy server uses a DNS server to get the address of Bob's proxy server, and then forwards the INVITE request to that server. The server then forwards the request to Bob's user agent, causing Bob's phone to ring.

The User agent Alice connects to the proxy server through LAN using invite, s i p colon bob at the rate biloxi dot com from s i p colon alice at the rate atlanta dot com. The proxy server connects to the server. The server returns I P address of Bob proxy server. The invite s i p colon bob at the rate biloxi dot com from s i p colon alice at the rate atlanta dot com is sent to another proxy server through internet. The proxy server sends the invite s i p colon bob at

the rate biloxi dot com from s i p colon alice at the rate atlanta dot com to user agent Bob through a wireless network.

INVITE of Death

The INVITE of Death vulnerability was discovered on February 16, 2009. The vulnerability allows the attacker to crash the server causing remote Denial of Service (DoS) by sending a single malformed packet. An impersonator can, using a malformed packet, overflow the specific string buffers, add a large number of token characters, and modify fields in an illegal fashion. As a result, a server is tricked into an undefined state, which can lead to call processing delays, unauthorized access, and a complete denial of service. The problem specifically exists in OpenSBC version 1.1.5-25 in the handling of the "Via" field from a maliciously crafted SIP packet.

For the popular, open source-based Asterisk PBX there are security advisories that cover not only signaling-related problems, but also problems with other protocols and their resolution. Problems may be malformed SDP attachments where codex numbers are out of the valid range or obsolete headers such as "Also".

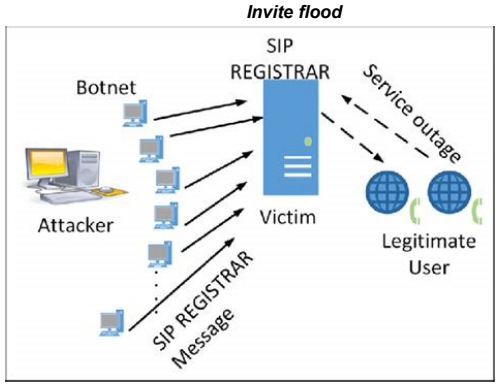
The INVITE of Death is specifically a problem for operators that run their servers on the public internet. Because SIP allows the usage of UDP packets, it is easy for an attacker to spoof any source address in the internet and send the INVITE of death from untraceable locations. By sending these kinds of requests periodically, attackers can completely interrupt the telephony service. The only choice for the service provider is to upgrade their systems until the attack does not crash the system anymore.

Read more about this topic: INVITE Of Death

Example App DoS: SIP Invite flood

Either SIP Invite or
SIP Registrar messages

Effect:
- Server crashes
- Network might congest
- User receives a zillion calls

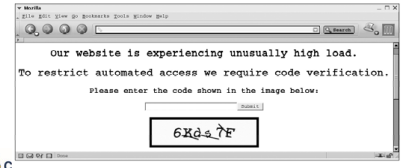


SIP Flood

A SIP flood attack exploits the fact that a single INVITE request triggers considerable resource consumption. The attacker can flood a SIP proxy with numerous INVITE requests with spoofed IP addresses, or alternately a DDoS attack using a botnet to generate numerous INVITE request. This attack puts a load on the SIP proxy servers in two ways. First, their server resources are depleted in processing the INVITE requests. Second, their network capacity is consumed. Call receivers are also victims of this attack. A target system will be flooded with forged VoIP calls, making the system unavailable for legitimate incoming calls.

Hypertext Transfer Protocol (HTTP) Based Attacks – CyberSlam

- Mimic legitimate Web browsing behavior and consume higher layer server resources such as CPU, memory, database and diskband width.
- Small amount of requests which take busy a WebServer towards the Databases with stressful queries
  - Recent FBI case from a web site competitor against another
- From the Web server's perspective, these zombie requests (from a botnet) look exactly like legitimate requests, so the server ends up spending a lot of its time serving the zombies, causing legitimate users to be denied service.

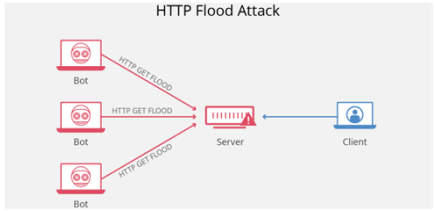


Potential solution: Smart Captcha sent only to human-like patterns

- Note: using captcha to query bots might not be enough to kill them, as a server might consume resources to generate thousands captcha

http://www.nms.lcs.mit.edu/~kandula/projects/killbots/kandula_login.pdf
http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-969.pdf

HTTP flood and recursive HTTP flood



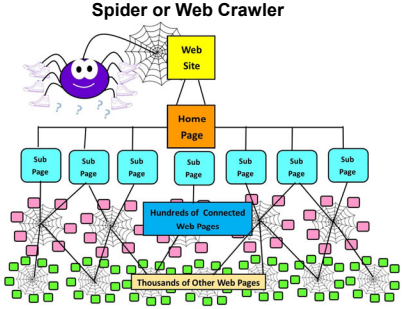
Based on HTTP GET and POST message types sent from a botnet.

Recursive variant is HTTP request sent to all the URLs linked inside the initial page

Example of hacker/stresser tools:
Low Orbit Ion Cannon (LOIC), High Orbit Ion Cannon (HOIC)

note

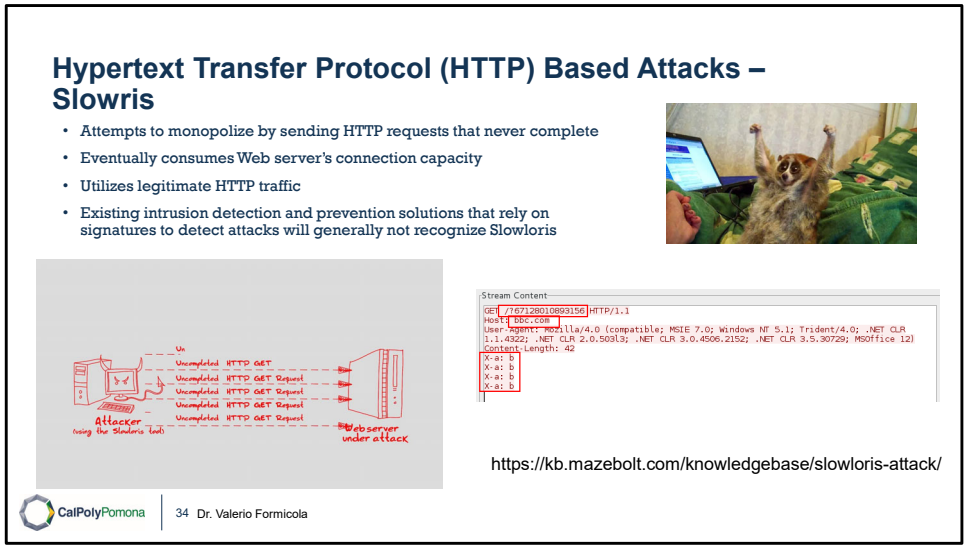
Spider or Web Crawler



A WebCrowler or Spider is a bot that extracts all the http links (hyperlinks) in a starting website (HTTP seed), with the intent to download all of them, but not to flood them in a DoS

An HTTP flood refers to an attack that bombards Web servers with HTTP requests. Typically, this is a DDoS attack, with HTTP requests coming from many different bots. The requests can be designed to consume considerable resources. For example, an HTTP request to download a large file from the target causes the Web server to read the file from hard disk, store it in memory, convert it into a packet stream, and then transmit the packets. This process consumes memory, processing, and transmission resources.

A variant of this attack is known as a recursive HTTP flood. In this case, the bots start from a given HTTP link and then follows all links on the provided Web site in a recursive way. This is also called spidering.



An intriguing and unusual form of HTTP-based attack is Slowloris [SOUR12], [DAMO12]. Slowloris exploits the common server technique of using multiple threads to support multiple requests to the same server application. It attempts to monopolize all of the available request handling threads on the Web server by sending HTTP requests that never complete. Since each request consumes a thread, the Slowloris attack eventually consumes all of the Web server's connection capacity, effectively denying access to legitimate users.

The HTTP protocol specification (RFC2616) states that a blank line must be used to indicate the end of the request headers and the beginning of the payload, if any. Once the entire request is received, the Web server may then respond. The Slowloris attack operates by establishing multiple connections to the Web server. On each connection, it sends an incomplete request that does not include the terminating newline sequence. The attacker sends additional header lines periodically to keep the connection alive, but never sends the terminating newline sequence. The Web server keeps the connection open, expecting more information to complete the request. As the attack continues, the volume of long-standing Slowloris

connections increases, eventually consuming all available Web server connections, thus rendering the Web server unavailable to respond to legitimate requests.

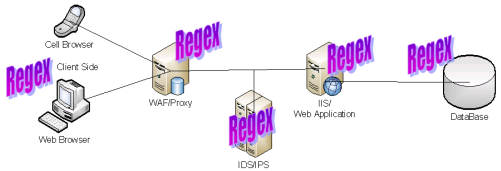
Slowloris is different from typical denials of service in that Slowloris traffic utilizes legitimate HTTP traffic, and does not rely on using special “bad” HTTP requests that exploit bugs in specific HTTP servers. Because of this, existing intrusion detection and intrusion prevention solutions that rely on signatures to detect attacks will generally not recognize Slowloris. This means that Slowloris is capable of being effective even when standard enterprise-grade intrusion detection and intrusion prevention systems are in place.

There are a number of countermeasures that can be taken against Slowloris type attacks, including limiting the rate of incoming connections from a particular host; varying the timeout on connections as a function of the number of connections; and delayed binding. Delayed binding is performed by load balancing software. In essence, the load balancer performs an HTTP request header completeness check, which means that the HTTP request will not be sent to the appropriate Web server until the final two carriage return and line feeds are sent by the HTTP client. This is the key bit of information. Basically, delayed binding ensures that your Web server or proxy will never see any of the incomplete requests being sent out by Slowloris.

Complexity DoS - Regular expression denial of service (ReDoS)

- The Regular expression Denial of Service (ReDoS) is a Denial of Service attack, that exploits the fact that most Regular Expression implementations may reach extreme situations that cause them to work very slowly (exponentially related to input size). An attacker can then cause a program using a Regular Expression (Regex) to enter these extreme situations and then hang for a very long time.

A Regex pattern is called **Evil Regex** if it can get stuck on crafted input.



Components that might contain Evil Regex can get hit by a sequence that hangs the node (e.g., a web server or an IDS)

Examples of Evil Regex:

(a+)+
([a-zA-Z]+)*
(a|aa)+
(a|a?)+
(.*a){x} for x \> 10

All the above are susceptible to the input aaaaaaaaaaaaaaaaaaaaaaaa!

https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS

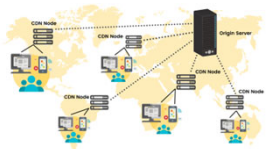
In every layer of the WEB there are Regular Expressions, that might contain an Evil Regex. An attacker can hang a WEB-browser (on a computer or potentially also on a mobile device), hang a Web Application Firewall (WAF), attack a database, and even stack a vulnerable WEB server.

For example, if a programmer uses a Regex to validate the client side of a system, and the Regex contains an Evil Regex, the attacker can assume the same vulnerable Regex is used in the server side, and send a well-crafted input, that stacks the WEB server.


Defensive measures
against DoS
CalPolyPomona

Observation: Flashcrowd events

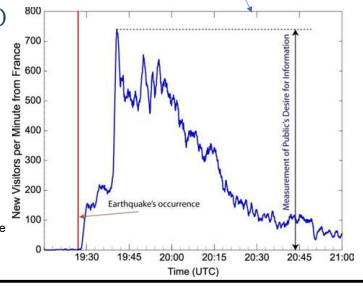
- High traffic volumes may be legitimate
  - High publicity about a specific site
  - Activity on a very popular site
  - Described as **slashdotted, flash crowd, or flash event**
  - Solutions are **Load Balancers** in datacenters And **Content Delivery Networks (CDN)** (e.g., Akamai)



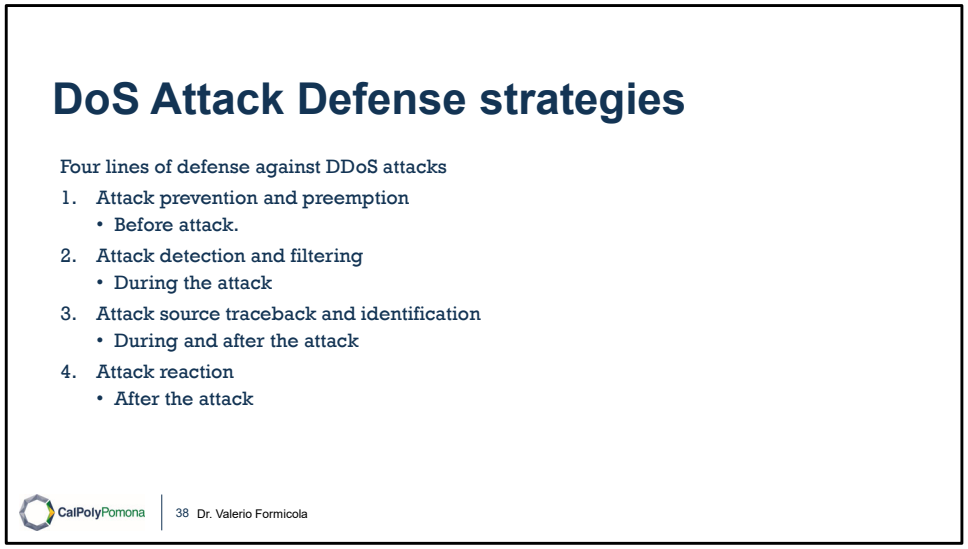
Example:
- at 7 PM Amazon will offer product 50% of price on a specific day
- world cup final free to watch at website X
- some natural/human related event occurring, and everybody connects to the most reliable info source

CalPolyPomona | 37 Dr. Valerio Formicola

Access to the website of EMSC - European-Mediterranean Seismological Centre after an earthquake somewhere in Europe



Sometime network congestion is not malicious. sSmetimes occurs by accident as a result of high publicity about a specific site. Classically, a posting to the well-known Slashdot news aggregation site often results in overload of the referenced server system. Similarly, when popular sporting events like the Olympics or Soccer World Cup matches occur, sites reporting on them experience very high traffic levels. This has led to the terms *slashdotted , flash crowd , or flash event being used to describe* such occurrences. There is very little that can be done to prevent this type of either accidental or deliberate overload without also compromising network performance. The provision of significant excess network bandwidth and replicated distributed servers is the usual response, particularly when the overload is anticipated. This is regularly done for popular sporting sites. However, this response does have a significant implementation cost. Example of strategies to allocate dynamic resources is to use CDN technologies



There are a number of steps that can be taken both to limit the consequences of being the target of a DoS attack and to limit the chance of your systems being compromised and then used to launch DoS attacks. It is important to recognize that these attacks cannot be prevented entirely.

In general, there are four lines of defense against DDoS attacks [PENG07, CHAN02]:

• **Attack prevention and preemption (before the attack):** These mechanisms enable the victim to endure attack attempts without denying service to legitimate clients. Techniques include enforcing policies for resource consumption and providing backup resources available on demand. In addition, prevention mechanisms modify systems and protocols on the Internet to reduce the possibility of DDoS attacks.

• **Attack detection and filtering (during the attack):** These mechanisms attempt to detect the attack as it begins and respond immediately. This minimizes the impact of the attack on the target. Detection involves looking for suspicious patterns of behavior. Response involves filtering out packets likely to be part of the

attack.

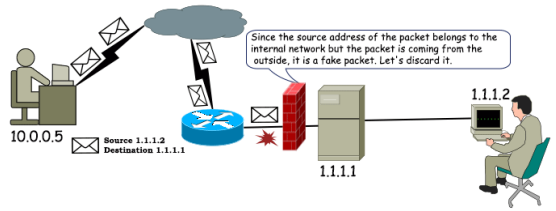
• **Attack source traceback and identification (during and after the attack):** This is an attempt to identify the source of the attack as a first step in preventing future attacks. However, this method typically does not yield results fast enough, if at all, to mitigate an ongoing attack.

• **Attack reaction (after the attack):** This is an attempt to eliminate or curtail the effects of an attack.

We discuss the first of these lines of defense in this section and consider the remaining three in Section 7.7 .

DoS Attack Prevention – Spoofed IP filtering

- Block spoofed source addresses, i.e., drop packets before they leave the spoofed machine:
  - On routers as close to source as possible, example with "Reverse Path Filtering" strategy: https://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.kernel.rpf.html
  - An ISP (Internet Service Provider) should filter out all packets it is known they are not generated within the possible assigned IP addresses (an ISP knows what is assigned since it is its own network)



CalPolyPomona | 39 Dr. Valerio Formicola

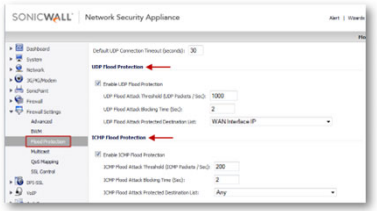
A critical component of many DoS attacks is the use of spoofed source addresses. These either obscure the originating system of direct and distributed DoS attacks or are used to direct reflected or amplified traffic to the target system. Hence one of the fundamental, and longest standing, recommendations for defense against these attacks is to limit the ability of systems to send packets with spoofed source addresses. RFC 2827, *Network Ingress Filtering: Defeating Denial-of-service attacks which employ IP Source Address Spoofing , directly makes this recommendation, as* do SANS, CERT, and many other organizations concerned with network security.

This filtering needs to be done as close to the source as possible, by routers or gateways knowing the valid address ranges of incoming packets. Typically this is the ISP providing the network connection for an organization or home user. An ISP knows which addresses are allocated to all its customers and hence is best placed to ensure that valid source addresses are used in all packets from its customers. This type of filtering can be implemented using explicit access control rules in a router to ensure that the source address on any customer packet is one allocated to the ISP.Alternatively, filters may be used to ensure that the path

back to the claimed source address is the one being used by the current packet. For example, this may be done on Cisco routers using the “ip verify unicast reverse-path” command. This latter approach may not be possible for some ISPs that use a complex, redundant routing infrastructure. Implementing some form of such a filter ensures that the ISP’s customers cannot be the source of spoofed packets. Regrettably, despite this being a well-known recommendation, many ISPs still do not perform this type of filtering. In particular, those with large numbers of broadband-connected home users are of major concern. Such systems are often targeted for attack as they are often less well secured than corporate systems. Once compromised, they are then used as intermediaries in other attacks, such as DoS attacks. By not implementing antispoofing filters, ISPs are clearly contributing to this problem. One argument often advancedfor not doing so is the performance impact on their routers. While filtering does incur a small penalty, so does having to process volumes of attack traffic. Given the high prevalence of DoS attacks, there is simply no justification for any ISP or organization not to implement such a basic security recommendation.

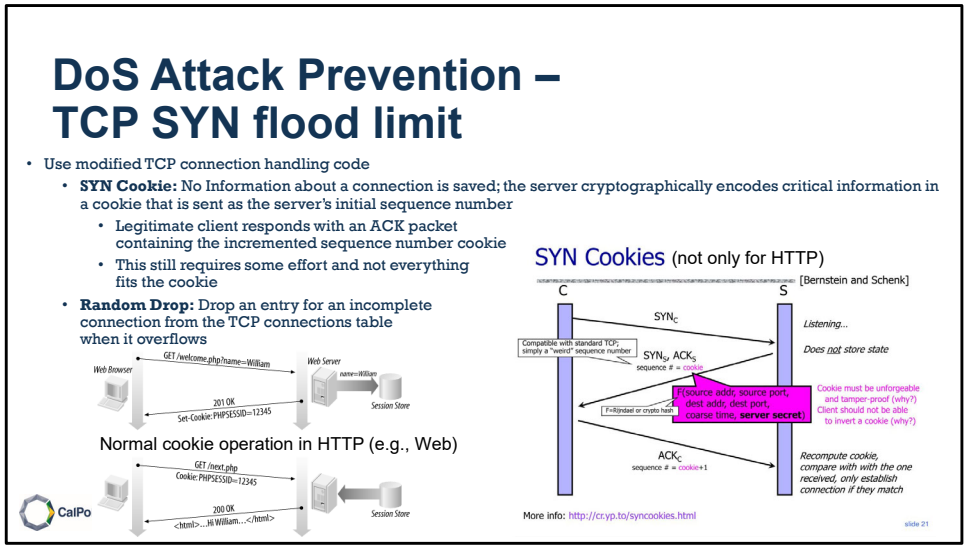
DoS Attack Prevention – ICMP and UDP throttling

- Attacks using particular packet types, such as ICMP floods or UDP floods to diagnostic services, can be throttled by imposing limits on the rate at which these packets will be accepted. In normal network operation, these should comprise a relatively small fraction of the overall volume of network traffic



Network Throttling = Transmission Rate-limiting

Any defenses against flooding attacks need to be located back in the Internet cloud, not at a target organization's boundary router, since this is usually located after the resource being attacked. The filters must be applied to traffic before it leaves the ISP's network, or even at the point of entry to their network. While it is not possible, in general, to identify packets with spoofed source addresses, the use of a reverse path filter can help identify some such packets where the path from the ISP to the spoofed address differs to that used by the packet to reach the ISP. Also, attacks using particular packet types, such as ICMP floods or UDP floods to diagnostic services, can be throttled by imposing limits on the rate at which these packets will be accepted. In normal network operation, these should comprise a relatively small fraction of the overall volume of network traffic. Many routers, particularly the high-end routers used by ISPs, have the ability to limit packet rates. Setting appropriate rate limits on these types of packets can help mitigate the effect of packet floods using them, allowing other types of traffic to flow to the targeted organization even should an attack occur.



It is possible to specifically defend against the SYN spoofing attack by using a modified version of the TCP connection handling code. Instead of saving the connection details on the server, critical information about the requested connection is cryptographically encoded in a cookie that is sent as the server's initial sequence number. This is sent in the SYN-ACK packet from the server back to the client. When a legitimate client responds with an ACK packet containing the incremented sequence number cookie, the server is then able to reconstruct the information about the connection that it normally would have saved in the known TCP connections table. Typically this technique is only used when the table overflows. It has the advantage of not consuming any memory resources on the server until the three-way TCP connection handshake is completed. The server then has greater confidence that the source address does indeed correspond with a real client that is interacting with the server.

There are some disadvantages of this technique. It does take computation resources on the server to calculate the cookie. It also blocks the use of certain TCP extensions, such as large windows. The request for such an extension is normally saved by the server, along with other details of the requested connection.

However, this connection information cannot be encoded in the cookie as there is not enough room to do so. Since the alternative is for the server to reject the connection entirely as it has no resources left to manage the request, this is still an improvement in the system's ability to handle high connection-request loads. This approach was independently invented by a number of people. The best-known variant is **SYN Cookies,** whose principal originator is Daniel Bernstein. It is available in recent FreeBSD and Linux systems, though it is not enabled by default. A variant of this technique is also included in Windows 2000, XP, and later. This is used whenever their TCP connections table overflows.

Alternatively, the system's TCP/IP network code can be modified to selectively drop an entry for an incomplete connection from the TCP connections table when it overflows, allowing a new connection attempt to proceed. This is known as *selective drop or random drop* . On the assumption that the majority of the entries in an overflowing table result from the attack, it is more likely that the dropped entry will correspond to an attack packet. Hence its removal will have no consequence. If not, then a legitimate connection attempt will fail, and will have to retry. However, this approach does give new connection attempts a chance of succeeding rather than being dropped immediately when the table overflows.

Another defense against SYN spoofing attacks includes modifying parameters used in a system's TCP/IP network code. These include the size of the TCP connections table and the timeout period used to remove entries from this table when no response is received. These can be combined with suitable rate limits on the organization's network link to manage the maximum allowable rate of connection requests. None of these changes can prevent these attacks, though they do make the attacker's task harder.

DoS Attack Prevention – Amplification attacks

- Block IP **directed broadcasts**
- **Block suspicious services and combinations**
  - E.g., suspicious src IP and dst IP combinations
- Manage application attacks with a form of graphical puzzle **(captcha)** to distinguish legitimate human requests
  - Track also response time and modes, humans might behave differently than a bot
- Good general system security practices
  - Avoid to be infected and become a bot
- Use mirrored and replicated servers when high-performance and reliability is required
  - CDN and load balancing
- Cyber Threat intelligence:
  - **Darweb monitoring for Botnet services** available on the market

Captcha

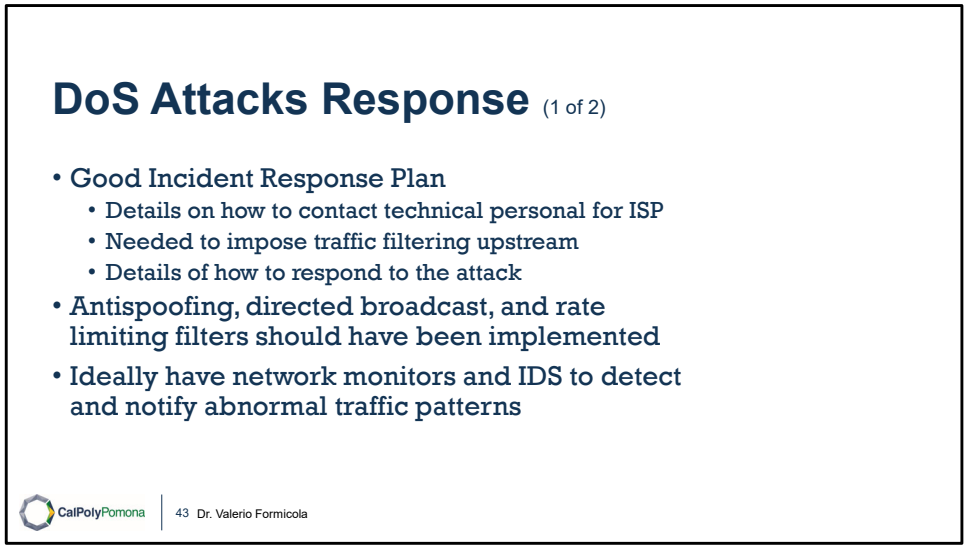
Botnet as a service

The best defense against broadcast amplification attacks is to block the use of IP-directed broadcasts. This can be done either by the ISP or by any organization whose systems could be used as an intermediary. As we noted earlier in this chapter, this and antispoofing filters are long-standing security recommendations that all organizations should implement. More generally, limiting or blocking traffic to suspicious services, or combinations of source and destination ports, can restrict the types of reflection attacks that can be used against an organization.

Defending against attacks on application resources generally requires modification to the applications targeted, such as Web servers. Defenses may involve attempts to identify legitimate, generally human initiated, interactions from automated DoS attacks. These often take the form of a graphical puzzle, a captcha, which is easy for most humans to solve but difficult to automate. This approach is used by many of the large portal sites like Hotmail and Yahoo. Alternatively, applications may limit the rate of some types of interactions in order to continue to provide some form of service. Some of these alternatives are explored in [KAND05].

Beyond these direct defenses against DoS attack mechanisms, overall good system security practices should be maintained. The aim is to ensure that your systems are not compromised and used as zombie systems. Suitable configuration and monitoring of high performance, well-connected servers is also needed to help ensure that they don't contribute to the problem as potential intermediary servers.

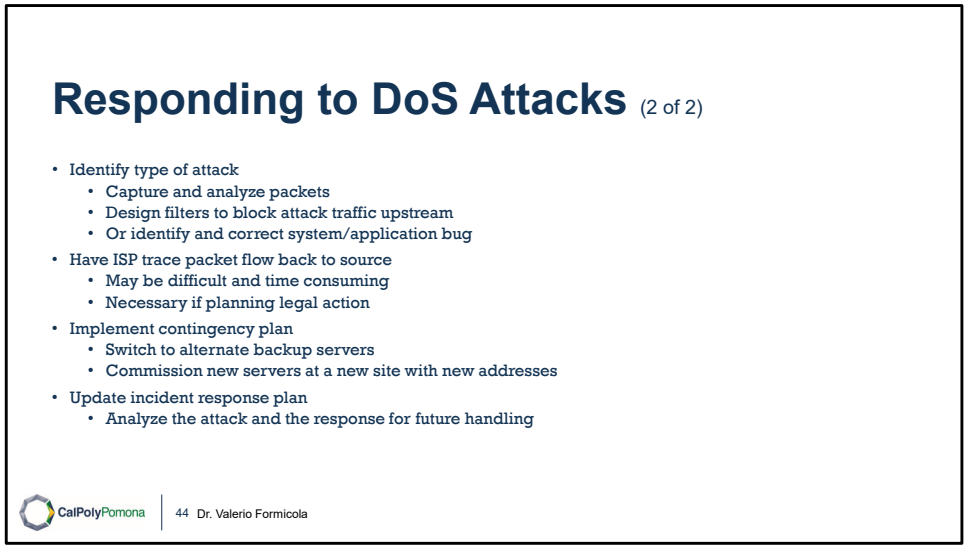
Lastly, if an organization is dependent on network services, it should consider mirroring and replicating these servers over multiple sites with multiple network connections. This is good general practice for high-performance servers, and provides greater levels of reliability and fault tolerance in general and not just a response to these types of attack.



To respond successfully to a DoS attack, a good incident response plan is needed. This must include details of how to contact technical personal for your Internet service provider(s). This contact must be possible using nonnetworked means, since when under attack your network connection may well not be usable. DoS attacks, particularly flooding attacks, can only be filtered upstream of your network connection. The plan should also contain details of how to respond to the attack. The division of responsibilities between organizational personnel and the ISP will depend on the resources available and technical capabilities of the organization.

Within an organization you should have implemented the standard antispoofing, directed broadcast, and rate limiting filters we discuss earlier in this chapter. Ideally, you should also have some form of automated network monitoring and intrusion detection system running so personnel will be notified should abnormal traffic be detected. We discuss such systems in Chapter 8. Research continues as to how best identify abnormal traffic. It may be on the basis of changes in patterns of flow information, source addresses, or other traffic characteristics, as [CARL06] discuss. It is important that an organization knows its normal traffic

patterns so it has a baseline with which to compare abnormal traffic flows. Without such systems and knowledge, the earliest indication is likely to be a report from users inside or outside the organization that its network connection has failed. Identifying the reason for this failure, whether attack, misconfiguration, or hardware or software failure, can take valuable additional time to identify.



When a DoS attack is detected, the first step is to identify the type of attack and hence the best approach to defend against it. Typically this involves capturing packets flowing into the organization and analyzing them, looking for common attack packet types. This may be done by organizational personnel using suitable network analysis tools. If the organization lacks the resources and skill to do this, it will need to have its ISP perform this capture and analysis. From this analysis the type of attack is identified, and suitable filters are designed to block the flow of attack packets. These have to be installed by the ISP on its routers. If the attack targets a bug on a system or application, rather than high traffic volumes, then this must be identified and steps taken to correct it and prevent future attacks.

The organization may also wish to ask its ISP to trace the flow of packets back in an attempt to identify their source. However, if spoofed source addresses are used, this can be difficult and time-consuming. Whether this is attempted may well depend on whether the organization intends to report the attack to the relevant law enforcement agencies. In such a case, additional evidence must be collected and actions documented to support any subsequent legal action.

In the case of an extended, concerted, flooding attack from a large number of distributed or reflected systems, it may not be possible to successfully filter enough of the attack packets to restore network connectivity. In such cases, the organization needs a contingency strategy either to switch to alternate backup servers or to rapidly commission new servers at a new site with new addresses, in order to restore service. Without forward planning to achieve this, the consequence of such an attack will be extended loss of network connectivity. If the organization depends on this connection for its function, the consequences on it may be significant.

Following the immediate response to this specific type of attack, the organization's incident response policy may specify further steps that are taken to respond to contingencies like this. This should certainly include analyzing the attack and response in order to gain benefit from the experience and to improve future handling. Ideally the organization's security can be improved as a result. We discuss all these aspects of incident response further in Chapter 17 .


Appendix
CalPolyPomona

ICMP-based DoS attacks