Denial of Service Attacks

The Problem

What is Security?

The standard definition: CIA

- Confidentiality
- Integrity
- ☞ Availability

Thus far, we've concentrated on confidentiality and integrity—but availability can be a problem, too. This is often called a *denial of service* (DoS) attack.

How to Attack Availability

Two approaches:

- Disrupt a system
- Overload a system

Constraints:

- Local or remote?
- Your resources or someone else's?
- Privileged or unprivileged?

Too Trivial to Matter?

```
# sudo rm -rf /
```

- Local access
- Requires privilege

Some Unprivileged Denial of Service Attacks

```
$ while true
do
    mkdir x
    cd x
done
```

or

```
$ while true
do
    yes This is a DoS Attack &
done
```

RAM and Paging

- All modern systems implement *virtual memory*—which, among other things, lets the computer pretend to have far more RAM than it really has
- This comes at a cost in performance
- When the active virtual memory/real memory ratio gets too high, the system starts *thrashing*
- If deliberate, it's an attack

Resources Attacked

- CPU time
- RAM
- Disk space or other file system components
- Files
- Network bandwidth
- Computer hardware!

All have been attacked

Limits

- To defend against these attacks, operating systems implement resource conumption limits
- Example: the `setrlimit()` and `quotactl()` system calls on Linux
- Example: the scheduler in the kernel allocates CPU time among processes

The Ulimit Command

```
$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 31505
max locked memory       (kbytes, -l) 65536
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 31505
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Viruses

- Most viruses were denial of service attacks
- Early ones were mostly harmless—but they annoyed people
- The first virus in the wild, Elk Cloner, displayed this message every 50th reboot

```
Elk Cloner: The program with a personality

It will get on all your disks
It will infiltrate your chips
Yes, it's Cloner!

It will stick to you like glue
It will modify RAM too
Send in the Cloner!
```

It attacked people's time!

Destructive Viruses

- Some viruses were destructive
- Example: CIH (AKA the Chernobyl Virus) would overwrite part of the disk and (on some PCs) part of the BIOS
- Result: loss of data and possibly functional loss of the computer itself
- (Note: both were often recoverable, with knowledge and effort)
- A denial of service attack against the disk and the hardware

The Role of Privilege

- Unprivileged programs can't launch some of these attacks
- This is why even single-person systems have user log in as non-root
- But a virus running with your privileges can delete your files
- (And it's why you shouldn't do casual stuff as `root`...)

Worms

- Unlike viruses (which, way back when, mostly spread by manual sharing of floppy disks), worms spread by themselves over the network
- They could have all of the effects of viruses—and they could consume network bandwidth
- Worms could launch denial of service attacks on the Internet

The SQL Slammer Worm

- Exploited a bug in Microsoft's SQL server
- Used UDP, not TCP — a single 376-byte packet to UDP port 1434 could infect a machine!
- Use of UDP instead of TCP let it spread much faster — one packet, from a forged source address, instead of a three-way handshake, payload transmission, and a three-packet `close()` sequence
- No direct damage, but it clogged network links very quickly

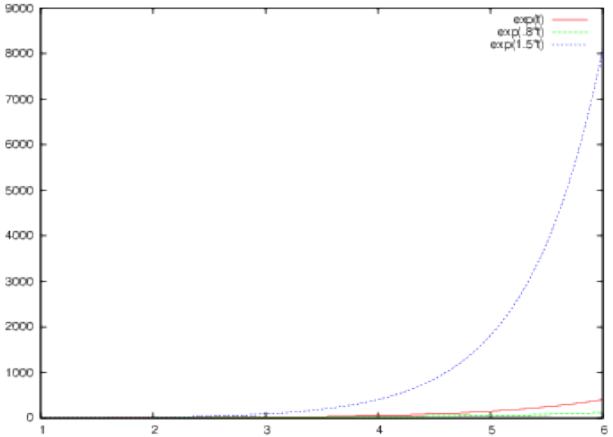
Denial of Travel Attack

- The Blaster worm shut down CSX Railroad's signaling network
- (You can run trains safely without signals, but at much lower capacity and with many more people involved)
- Blaster also shut down Air Canada's check-in terminals; Slammer affected Continental Airlines

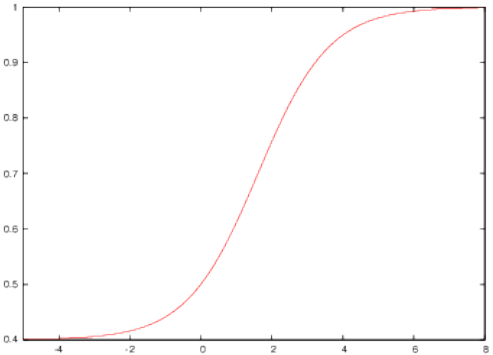
Spread Patterns

- Worms tend to exhibit *exponential growth* patterns
- They start slow, but get very big very quickly
- Equation: $y = e^{kt}$, where t is time
- If k is small, it spreads more slowly — but it still grows

Exponential Growth



There's a Ceiling



- Worms run out of vulnerable hosts
- Doesn't matter much if a machine is infected twice (and worms often prevent that)
- Actual graph is a *logistic curve*:

$$y = a\frac{1 + me^{-t/\tau}}{1 + ne^{-t/\tau}}$$

What Happened to DoS by Worm?

- The hackers have learned to make money from their evil deeds
- Spamming, phishing, credit card theft, ransomware, and more
- Taking down the Internet is bad for their business
- The Internet is up because the attackers want it to be. . .

Ransomware

- Penetrate a system or a site
- Encrypt user files
- If possible, encrypt backups, too
- Display a message
- When (if) the user pays (generally via Bitcoin or the like), they receive a decryption key
- (Well, sometimes they don't. . .)
- Denial of service for profit!

Cryptojacking: Malicious Cryptocurrency Mining

- Mining cryptocurrencies, e.g., Bitcoin, takes a lot of CPU power
- Hack into other folks' computers and let them do the mining for you
- Bonus: CPU consumption uses electricity, both directly and (often) for cooling—you steal their electricity, too
- A DoS attack on your CPU cycles and your wallet

Cryptojacking: Malicious Cryptocurrency Mining

- Mining cryptocurrencies, e.g., Bitcoin, takes a lot of CPU power
- Hack into other folks' computers and let them do the mining for you
- Bonus: CPU consumption uses electricity, both directly and (often) for cooling—you steal their electricity, too
- A DoS attack on your CPU cycles and your wallet
- Better variant: don't bother hacking; just buy web ads that contain JavaScript that does the mining

Cryptojacking: Malicious Cryptocurrency Mining

- Mining cryptocurrencies, e.g., Bitcoin, takes a lot of CPU power
- Hack into other folks' computers and let them do the mining for you
- Bonus: CPU consumption uses electricity, both directly and (often) for cooling—you steal their electricity, too
- A DoS attack on your CPU cycles and your wallet
- Better variant: don't bother hacking; just buy web ads that contain JavaScript that does the mining
- The legal variant: Salon Magazine used to ask non-subscribers' consent to load such JavaScript

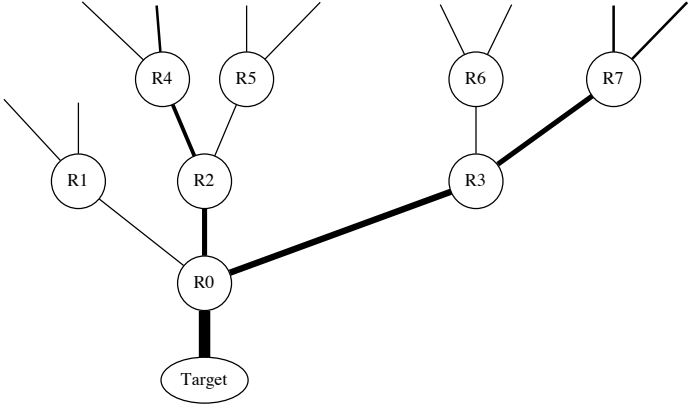
The Malign Power of the Internet

- Sometimes, an attacker doesn't have enough resources to launch a DoS attack
- The solution: use the Internet
- More precisely, use the power of many machines to attack a victim
- This is called a *distributed denial of sevice* (DDoS) attack

Simplest DDoS Attack

- Build a network of hacked computers
- Set up a *command-and-control* (C^2) network
- Send one command packet: “Flood this target”
- The C^2 network distributes the command to all of the “bots” (sometimes called “zombies”)
- All of the bots send traffic to the target, clogging its network link

The Effects of DDoS



The closer you are to the target, the more the links are clogged. The victim's link is almost completely overloaded.

Dodging Filters

- If the DDoS traffic came from fixed source addresses, the attack traffic could be filtered
- Countermove: use forged, random source addresses
- If the attack targeted a fixed port number, the attack traffic could be filtered
- Countermove: use random destination port numbers

DDoS for Mischief

- The early DDoS attacks were mostly for mischief
- In late 1999, in recovered source code, comments were found suggesting a planned attack at midnight, December 31, 1999
- Remember hearing about Y2K? Imagine if the Internet was down. . .

DDoS for Profit

- Warn a site that they'll be hit with a DDoS attack if they don't pay
- Launch a short attack to prove you can do it
- Especially vulnerable: sports gambling sites

DDoS for Political Reasons

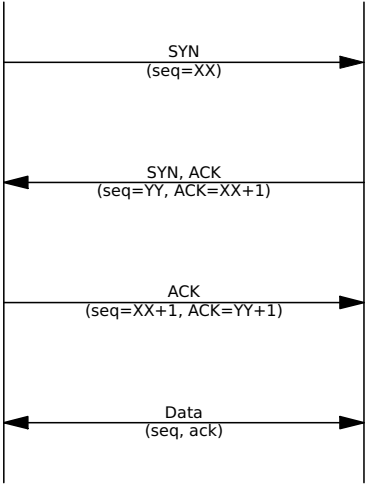
- Iran allegedly launched DDoS attacks against several U.S. banks
- “The attacks disabled victim bank websites, prevented customers from accessing their accounts online and collectively cost the victims tens of millions of dollars in remediation costs as they worked to neutralize and mitigate the attacks on their servers.”

Web-Based Flooding

- Create some JavaScript that repeatedly connects to a target site
- Tell your followers “click on this link to attack our enemies”
- If lots of folks do that, the target will be flooded with normal traffic

SYN Flooding

- Remember the TCP three-way handshake
- The client sends a SYN packet; the server replies with a SYN-ACK
- The client replies to the SYN-ACK with an ACK
- Until that third message is received, the server's side is half-open
- There is a limit to the number of half-open connections that are permitted
- If that number is exceeded, legitimate clients can't connect



The Linux `listen()` System Call

```
int listen(int sockfd, int backlog);
```

listen() marks the socket referred to by `sockfd` as a passive socket, that is, as a socket that will be used to accept incoming connection requests using **accept(2)**.

The `sockfd` argument is a file descriptor that refers to a socket of type **SOCK_STREAM** or **SOCK_SEQPACKET**.

The `backlog` argument defines the maximum length to which the queue of pending connections for `sockfd` may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of **ECONNREFUSED** or, if the underlying protocol supports retransmission, the request may be ignored so that a later reattempt at connection succeeds.

The `backlog` parameter controls the size of the queue

Reflector Attacks

- Find a UDP service where the response size is larger than the query—DNS is often a good choice, with (sometimes) a 10× ratio
- Send a query to that service, forging your source address to point to the victim
- The response—10× larger—will go to the victim
- Send these packets to lots of DNS servers—the victim will be hit with 10× more traffic than you sent

But What Do We Do?

Solutions

- DoS attacks are a *hard* problem—by definition, they're exhausting some resource
- Maybe you can increase your resources—but it's probably cheaper for the attacker to increase their nastiness
- But there are approaches

Distributed Services

- The essence of a DDoS attack is flooding one particular link
- Suppose you can spread out your site across many links and servers
- Either the attack hits one node and spares the others, if a single IP address is targeted, or the attack is diffused among many sites
- But how?

Content Distribution Networks

- Most large web sites use *Content Distribution Networks* (CDNs) to handle the load
- A CDN has mirror copies of the site on its worldwide nodes
- Big CDNs, like CloudFlare, have *many* nodes all over the world

The Really Big Sites

- Companies like Google and Amazon Web Services have *huge* pipes
- Google recently withstood a 2.5 Tbps—that is, $2.5 \cdot 10^{12}$ bits/second—DDoS attack
- Their services weren't affected at all...
- (They think it was a state-sponsored attack)
- Google's *Project Shield* puts journalists' sites on the Google cloud, to protect them from censorship-by-DDoS
- (But what if the DDoS attack is against network infrastructure, e.g., DNS servers?)

Filtering

- Sometimes, there are identifiable characteristics to the attack traffic
- Example: most legitimate traffic to a site is return visitors—during attacks, filter out unknown sources
- ISPs can filter such traffic before it reaches vulnerable links
- Or: filter out a single target IP address very early, at the edge of the network, to spare other addresses reached by that link
- The big ISPs know how to do this and have specialized filtering servers ready to go

Counterattack!

- Botnets are controlled by their C^2 networks
- If you take out the command nodes, the bots go idle
- This has been done...

Microsoft and Trickbot

- In October 2020, Microsoft got a court order allowing it to seize some IP addresses
- This let them take out most of the C^2 nodes in Trickbot, a million-node botnet
- (These bots were used for many sorts of evil, including launching ransomware attacks)
- At about the same time, Cyber Command—the US military's cyberwarfare unit—launched its own attack against Trickbot's C^2 nodes
- These attacks didn't destroy Trickbot, but it did knock it off the air for a while
- There have been hints that Trickbot was going to be used to disrupt the 2020 presidential election...

Restoring C^2 Networks

- The attackers have been building more resiliency into their tools
- They have backup communications channels and other ways to restore control
- Of course, the defenders try to block these, too

Cookies

- To defend against, e.g., SYN floods, the server can send a SYN+ACK packet with its sequence number created as a function of some client data
- (Invented by Dan Bernstein)
- It does *not* create state for the half-open connection—but when it receives the ACK (the third message in the three-way handshake), it can recognize and verify the ACK number as one it would have sent
- It then creates the state, after it knows that this isn't part of a SYN flood
- Some newer protocols have stronger versions of this built in

Client Puzzles Against CPU Exhaustion

- Some operatons, e.g., public key cryptography, take a lot of CPU time
- Attack: open lots of HTTPS connections, but don't do any expensive operations; just send fake data
- The server will detect this, but only after it's done its computations
- Solution: the server sends a hard-to-compute, cheap to verify "puzzle" to the client; it does not do anything expensive until it hears back
- Adjust the puzzle difficulty depending on CPU load; make the puzzle more expensive than the public key operations

A Sample Puzzle

- Pick a large, random number x
- Calculate $H(x)$, where H is a cryptographic hash function such as SHA512
- Set the low-order n bits of x to 0, where n is the difficulty parameter; call this x'
- Send the client $\langle x', H(x), n \rangle$
- The client must find x, but this cannot be done any more quickly than $O(2^n)$ time by exhaustive search
- Verification is cheap: one invocation of H on the returned value
- (There are better puzzles that are not as easily solved by using a massively parallel system, i.e., a botnet)

Designing for Defense

- Suppose you're designing a system—what do you do about DoS attacks?
- Protocols
- Parallelism
- Agility
- Planning
- (Cisco's advice is pretty good)

Protocols

- Make the client commit first—don't do expensive things until you're sure the client is real
- Stay stateless—package up all client state, encrypt+MAC it, and send it to the client. The client will return it with its next request, but you don't have to store anything
- If appropriate, use client-side puzzles

Parallelism

- Design your service to be split across multiple machines
- (You want to do that anyway—if you're successful, you'll need that to grow, and who wants to plan on being unsuccessful?)
- Don't assume low latencies between the different instances—assume that they may be all over the world

Cloud Providers

- If your service is cloud-based and distributed around the world, it's harder to attack
- The big cloud providers also have application tools to simplify splitting things up

Agility

- Monitor for attacks
- Be ready to move *quickly* if there's an attack
- Spin up more servers, move them around, work with a cloud provider
- Change IP addresses (and DNS records)

Planning

- Once the attack starts, you won't have time to fix your system architecture
- Plan ahead
  - Know whom to call at your ISP and/or cloud provider
  - Have those phone numbers written down—you may not be able to use email or text messages
- Figure out backup communications channels
- Rehearse your response—make sure you have all of the pieces in place
- DoS attacks are difficult to counter—but it is possible to do better than simply giving up

Questions?

(Tufted titmouse, Riverside Park, March 2, 2019)