ECE560
Computer and Information Security

Fall 2024

Denial of Service Attacks

Tyler Bletsch
Duke University

Definition

Denial-of-Service (DOS) Attack:

"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."

– NIST Computer Security Incident Handling Guide

Definition

- Attacks Availability (the “A” part of the CIA triad)

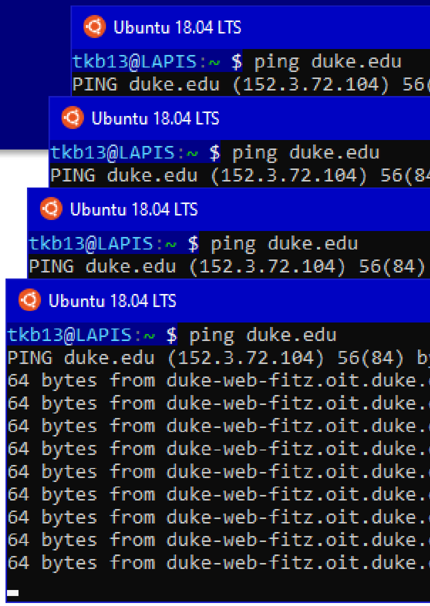
- Common types of resources targeted:
  - Network bandwidth (organizations have limited size network pipes)
  - System resources (CPU, memory, etc.)
  - Application resources (Connections, objects, file handles, etc.)

  - But there’s more…

Anything can be a resource

- Be careful in your thinking about DoS attacks
- May be tempted to think "DoS" = "network flood of some kind"
- DoS attacks, more generally, can attempt to exhaust *any* resource
- Things that are resources that you might not think of:
  - **Threads in a thread pool**: If a server has a capped or constant number of threads, getting them to service your requests, even if the threads are blocked, is a DoS attack (i.e., can tie up a server even when CPU is at 0%).
  - **Memory**: If your read function allocates memory "as needed", then all an attacker needs to do to knock you out is have you *need* to allocate unlimited memory (e.g. a 1TB URL).
  - **Random entropy**: `cat /dev/random` is a DoS attack on kernel entropy.
  - **ID numbers**: If each widget has a 16-bit ID number, then making 64k widgets is a DoS attack.

Classic DOS attack: Ping flood



Highly sophisticated cyber attack ->

- Ping flooding
  - Send lots of ICMP Ping packets
  - Default endpoint policy: Reply with echo packets
  - Default network policy: Treat all packets as equal, drop some when strained

    ^ bad

  - Better endpoint policy: Limit echo replies (rate, quantity, etc.)
  - Better network policy: Quality of Service (QoS) settings to deprioritize pings

    ^ better!

  - By default, the source of the attack is revealed, unless they **spoof the source address**

Source address spoofing

- Use a *forged* source address
  - Not allowed by OS by default, but can use a raw socket interface to craft your own packets (that are full of lies)
- Harder to identify attacking system
- Types of spoofing:
  - Claim to be a different machine **on your subnet**
    - Always works, hard to detect, but doesn't deflect your identity very far
  - Claim to be a machine on a **different subnet entirely**!
    - Deflects your identity to anyone on the internet! If it works…
    - Requires that routers not ask any questions as to why a packet from subnet X is coming from subnet Y
    - Well-configured routers would drop such packets
    - But that's extra work, since routers usually don't look at the source address at all
    - Result: too many networks are not well configured in this way ☹

Example

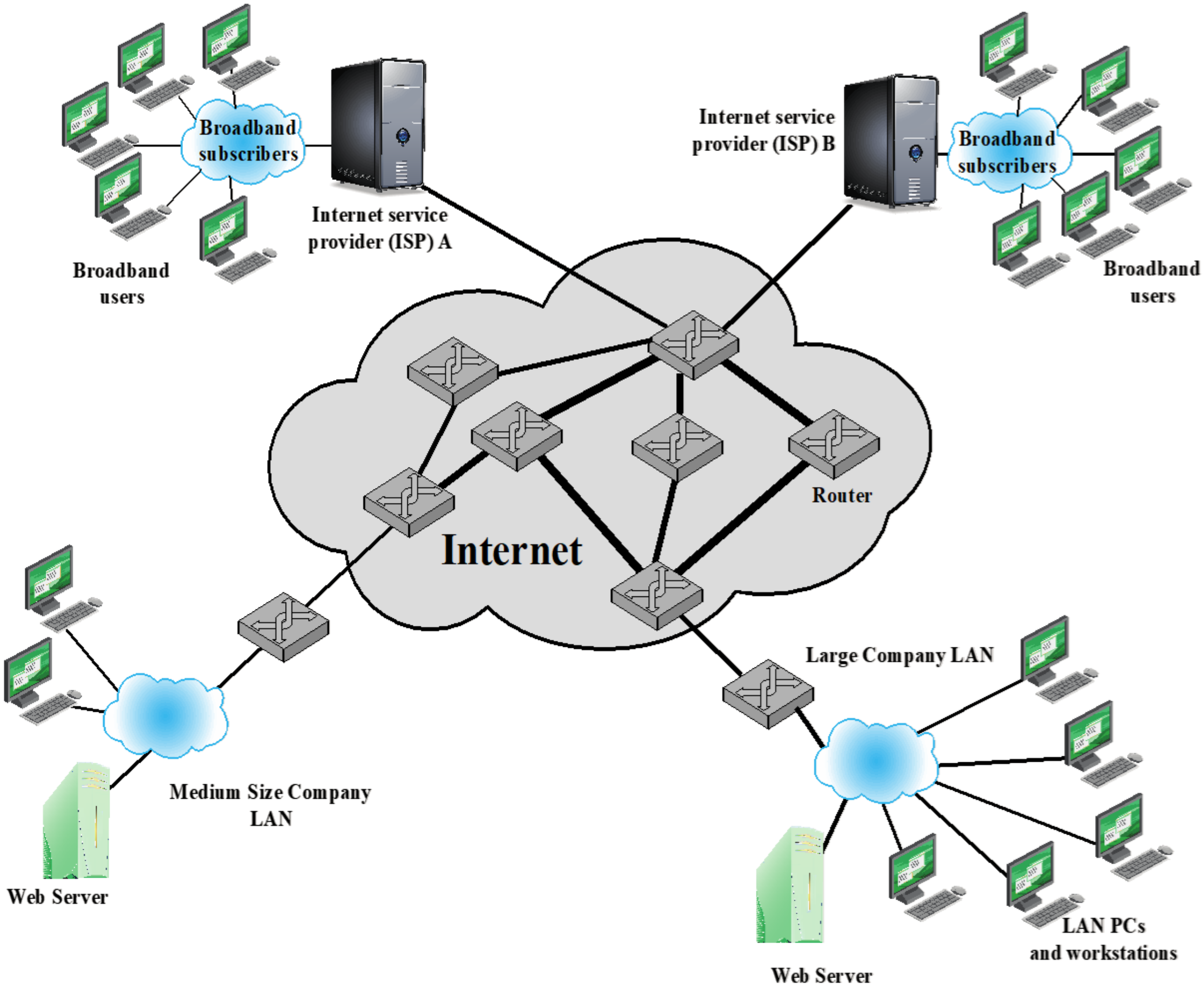


Figure 7.1 Example Network to Illustrate DoS Attacks

SYN Spoofing

- TCP three way handshake: SYN, ACK, SYN+ACK
- Server *receives* a SYN? Allocate lots of resources to handle the incoming connection (buffers, counters, table entries, etc.).
- Client *sends* a SYN? Normally, client OS allocates same structures.
- But what if you send a SYN but don't really mean it?
  - The OS isn't actually allocating resources for the outgoing connection!
- Result:
  - Cheap for attacker to send SYN packets
  - Expensive for receiver to handle them!

- Fills network connection table of server with little consequence to attacker!

SYN spoofing illustrated

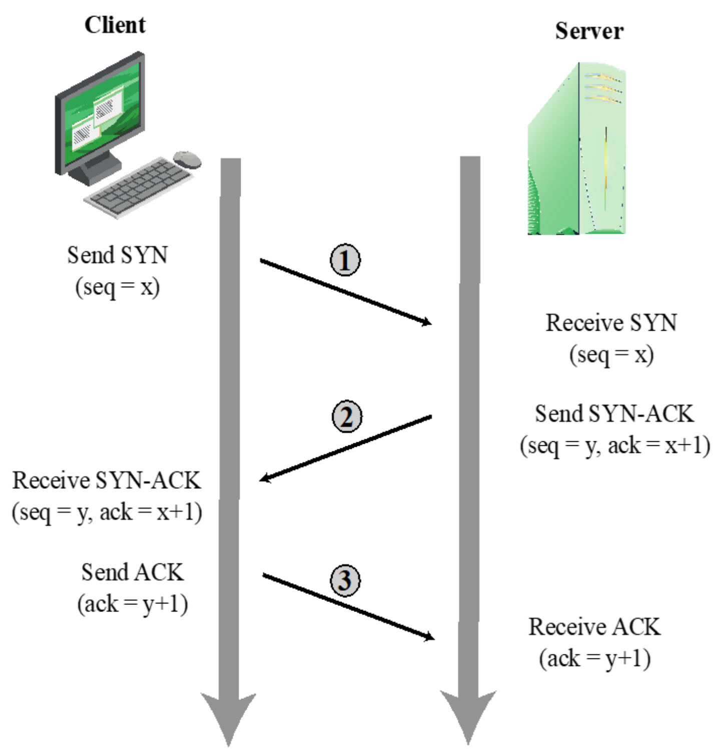


Figure 7.2 TCP Three-Way Connection Handshake

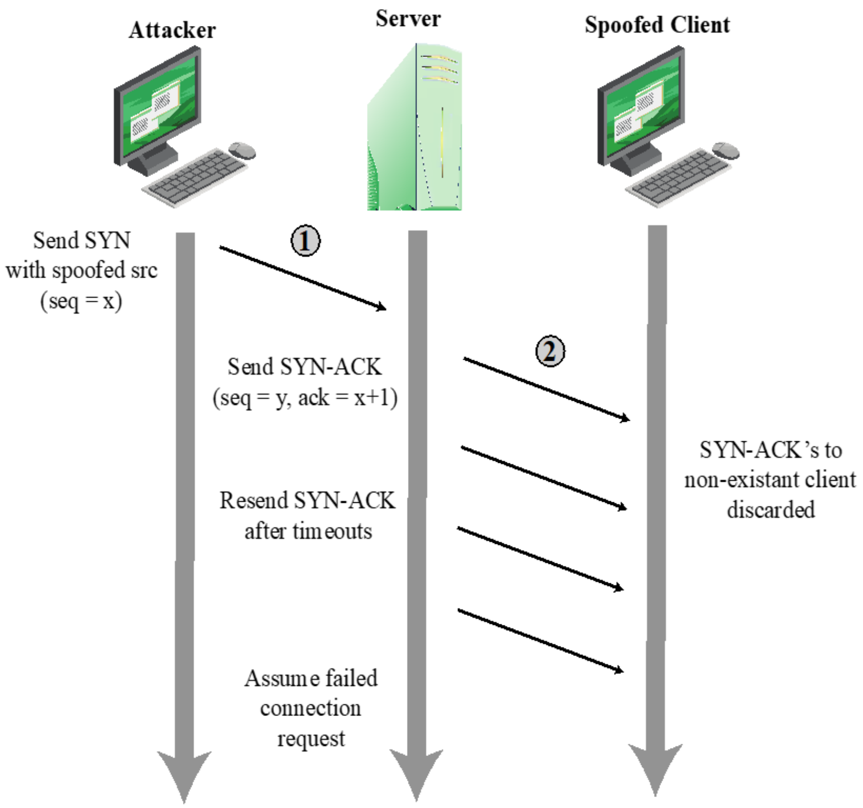


Figure7.3 TCP SYN Spoofing Attack

Other flooding attacks

- Trying to just fill up bandwidth?
- Can flood with any kind of packet, really. Not just ping.
- Examples:
  - **ICMP Ping** (covered earlier)
  - **Other ICMP packets** (traceroute, destination unreachable etc.): may need client permission, may be hard to filter out safely
  - **UDP**: easy to launch, no flow control, no client permissions needed
  - **TCP connect** (via OS socket interface): easy to launch, but uses OS resources
  - **TCP SYN**: needs client permission, expensive for receiver and cheap for sender

Distributed Denial of Service (DDOS) attacks

- More clients = better attack

- Where to get clients? Compromised machines!

- Use a worm or other attack to compromise a bunch of machines
- Install remote control software
  - **Zombies** or **bots** make up a **botnet**.
- Order them all to blast packets at a victim

DDOS Architecture

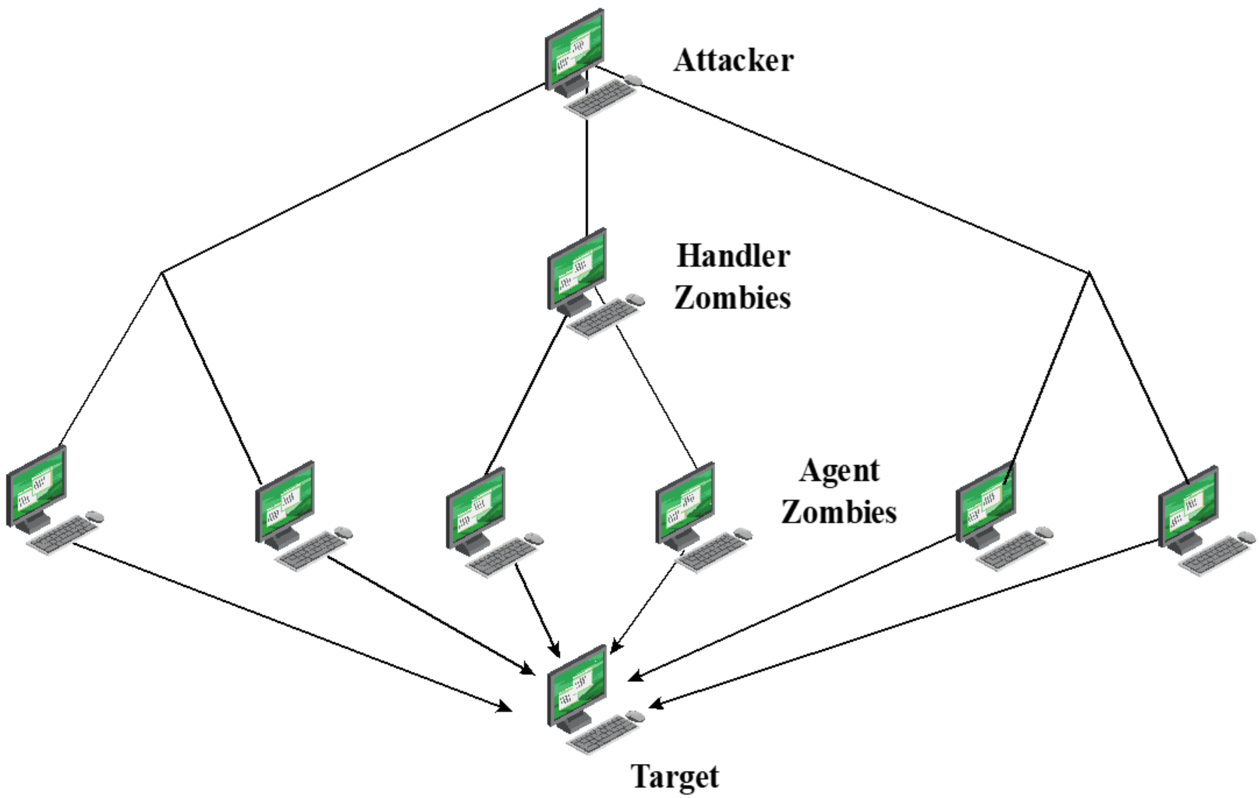
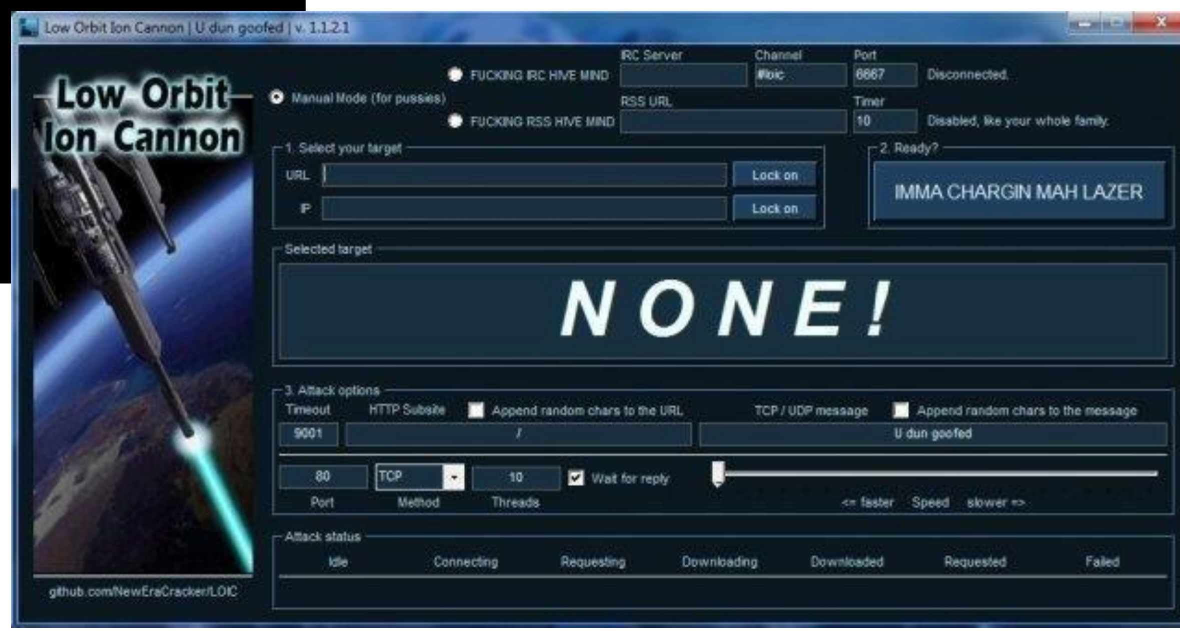


Figure 7.4 DDoS Attack Architecture

Not all zombies are victims



Hypertext Transfer Protocol (HTTP) Based Attacks

HTTP flood

- Same as any other flood
- Worse if the server has to do computation to respond. Contrast:
  - Visiting google.com
    vs
  - Doing a google *search*
- **Spidering**
  - Recursively visiting all the links on a site, so each visit is unique.

Slowloris

- Sending HTTP requests that never complete
- Consumes Web server's connection capacity with legitimate HTTP traffic
- Harder to detect – doesn't spike network throughput graphs, logs show results that look legitimate

Reflection Attacks

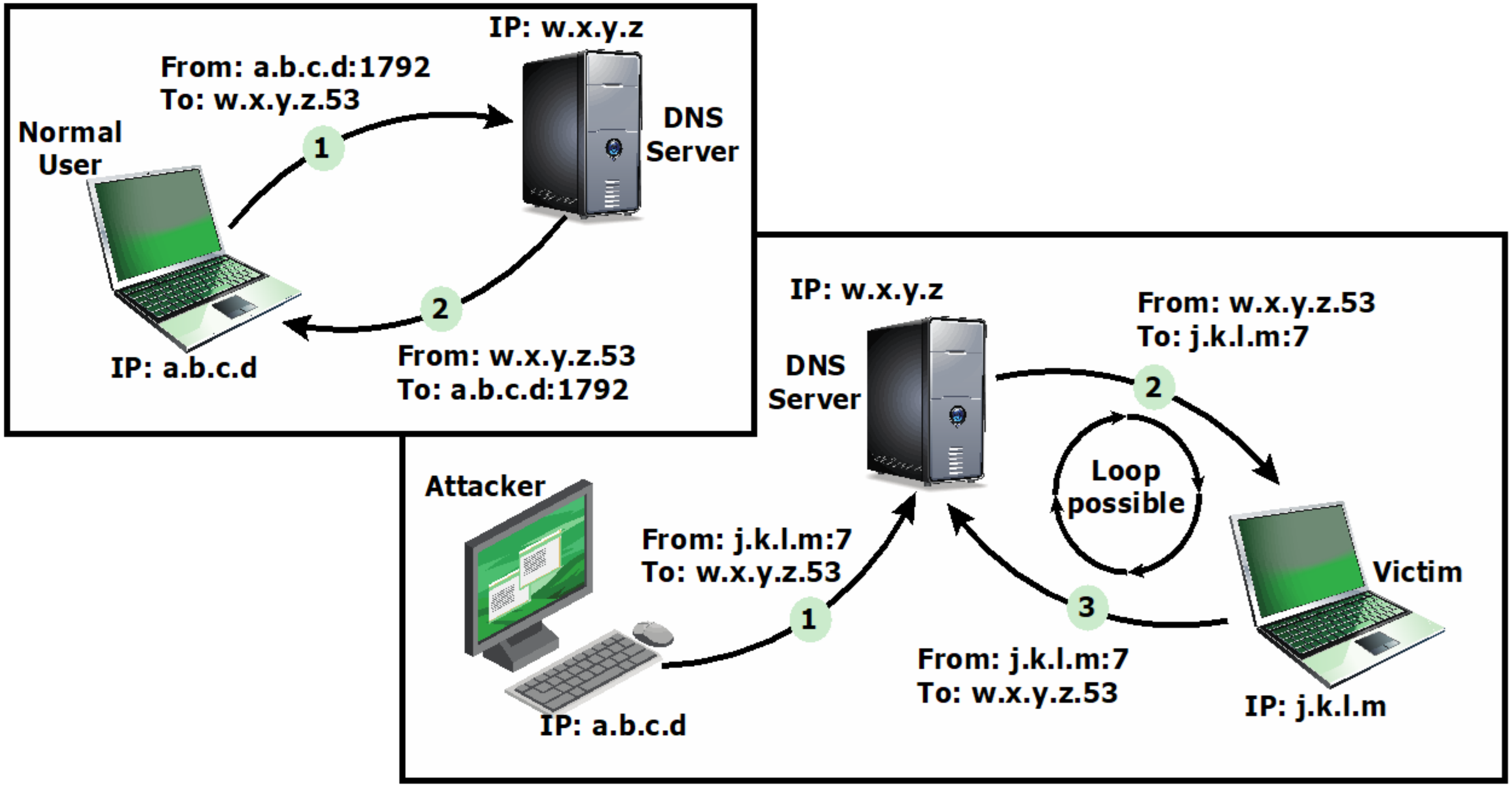
- Attacker sends packets to a known service on an **intermediary** with a **spoofed source address** of the *actual* **target system**
  - Intermediary responds; response sent to the target

- In effect, we "reflect" the attack off the intermediary (reflector)
  - **Amplification**: Attack is more effective if the reflection is bigger than the original request

- Goal: generate enough traffic to flood the link to the target system (ideally without alerting the intermediary)

- Defender solution: Same as other spoofing attacks – why are networks allowing spoofed-source packets to go out???

Simple reflection attack based on the old "echo" service



Note: this example uses port 7 (echo), which nobody has on any more, because of this attack and others like it.

Figure 7.6 DNS Reflection Attack

Amplification example: DNS amplification

- DNS **requests** can be *small* ("tell me about google.com")
- DNS **responses** can be *large* (all the google.com DNS records)
- Spoof source on little DNS requests to many public DNS servers, they send big responses to the spoofed source
- Can magnify attacker bandwidth by 50x!

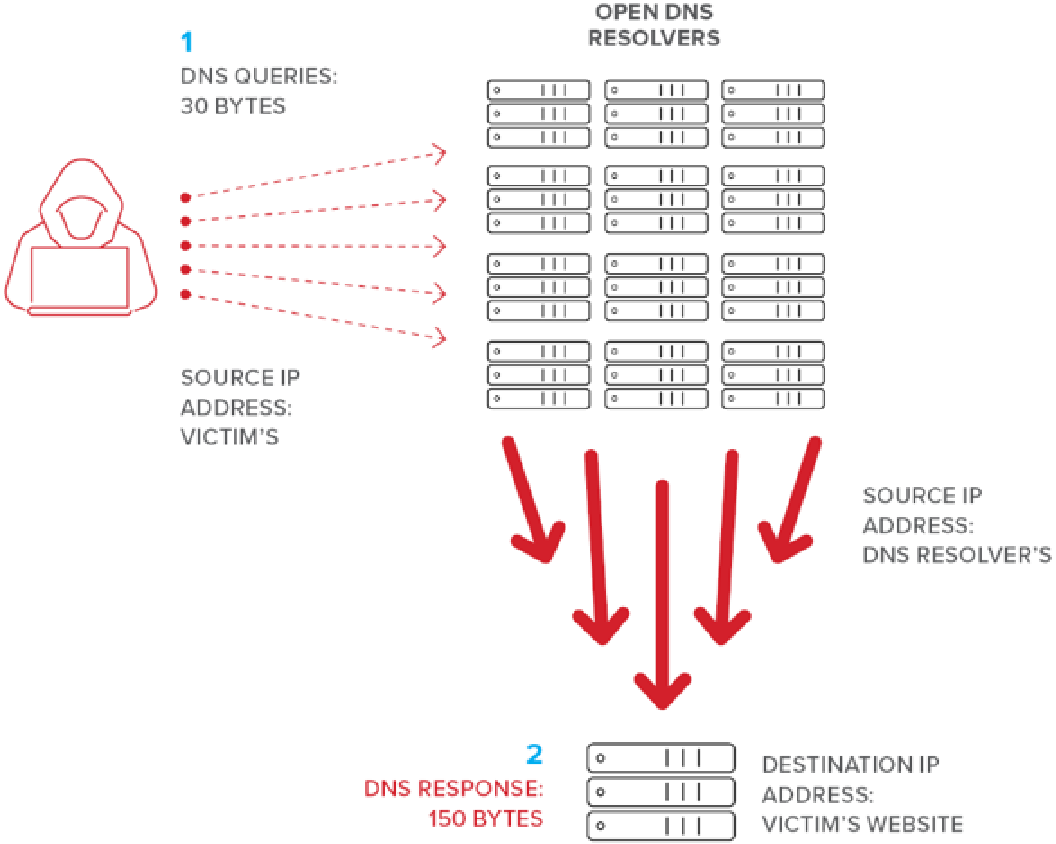
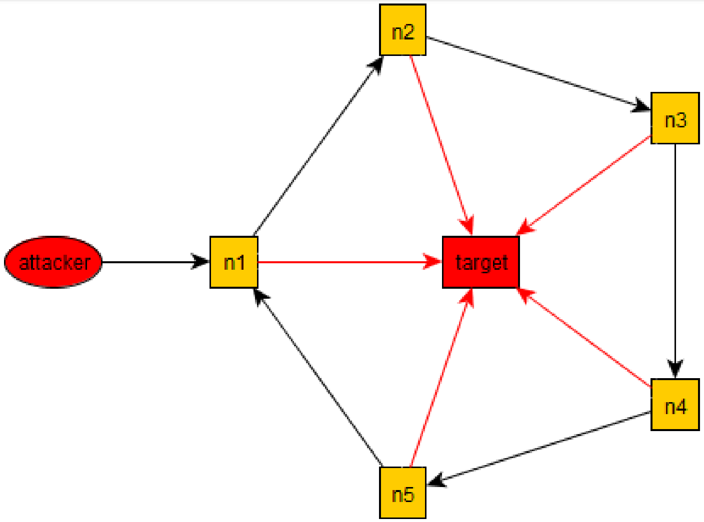


Figure source

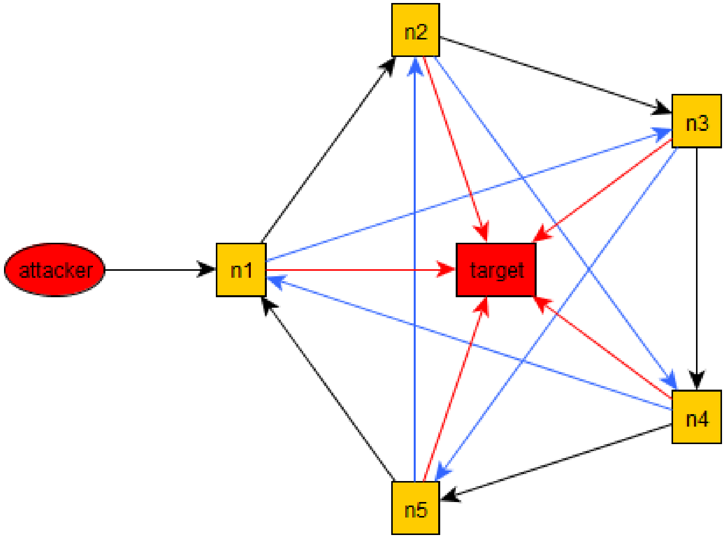
Cyclic amplification

- If a service can be made to forward to 2 targets, a loop can be formed that attacks a target at each iteration (constant rate)



- If a service forwards to 3+ targets, the loop can attack & grow (exponential rate)

Example:
e-mail forwarding
systems



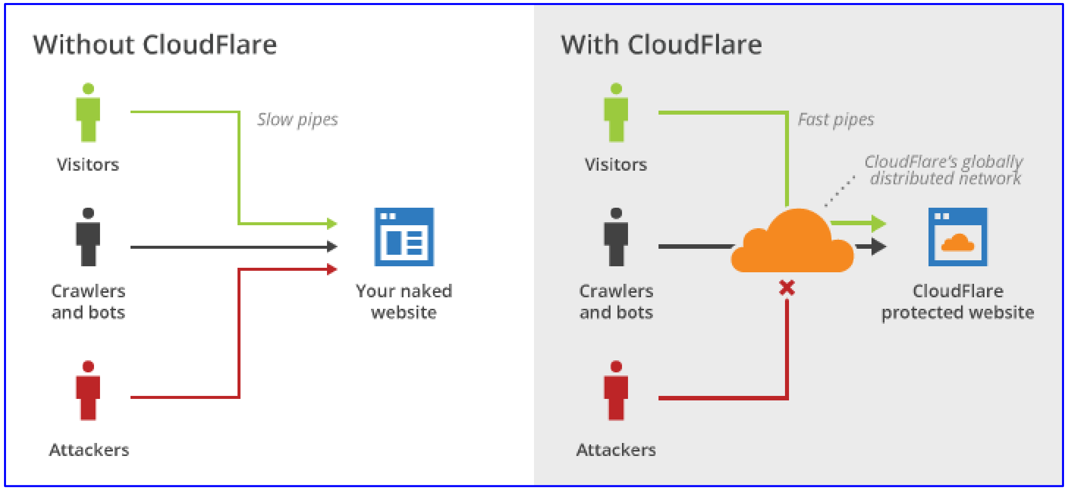
DOS defenses: prevention

- Prevention through configuration
  - Block spoofed source addresses from your network (helps others)
  - Block IP directed broadcasts (the ability to send to 1.2.3.*)
  - Disable needless services
  - Rate limit certain traffic upstream (e.g., max ICMP pings per second)
- Prevention through specific tricks
  - TCP: Encode connection info in sequence number,
    only allocate buffers on SYN+ACK (step 3 of connection instead of step 1)
  - TCP: If connection table overflowing, drop a random “awaiting SYN+ACK” one
  - Interactive service: Require 

    on repeated/heavy load
- Prevention through money
  - Have additional servers on standby (either physically or via cloud)
  - Pay someone with a huge cloud to front-end your services (e.g. CloudFlare)

Example: Website protection with CloudFlare (or similar services)

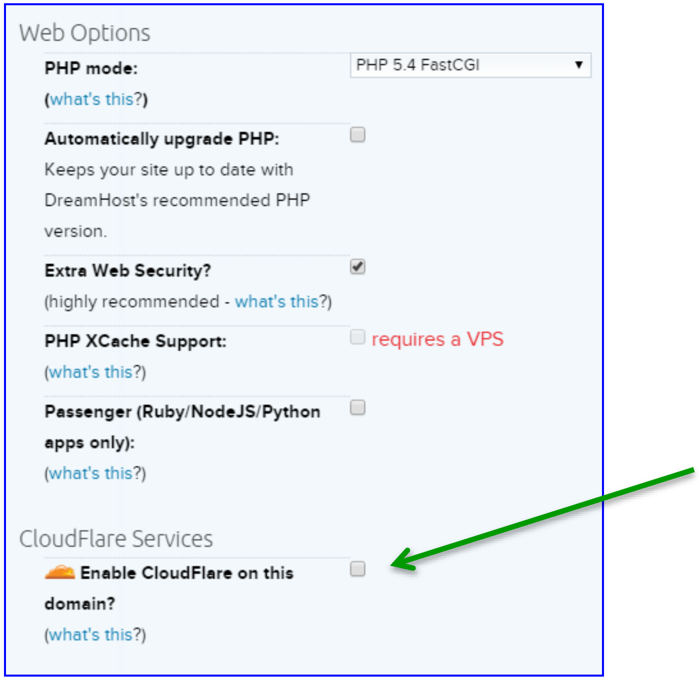
- General idea: pay someone else to absorb the DDOS and filter it. (Often free for small sites.)



Here's a diagram so high-level and fluffy so as to make it useless.

Example: Website protection with CloudFlare (or similar services)

- Some web hosts offer it as a one-click option.
- If not, it's just a matter of changing DNS settings so stuff gets handled by CloudFlare before hitting your server



Build in site settings on a popular webhost

DOS defenses: response

- Have a **response plan**
  - Need to get ***upstream*** connection to block malicious traffic: have contact info for ISP, especially via non-internet means!
  - **Identify** type of attack (capture packets, analyze what you find)
  - **Design filters** that will block just the attack traffic
    - What characteristics about it are unique? Same source, same content?
    - Tell your ISP
  - Have a **backup deployment plan**
    - Deploy new servers, change addresses, etc.

Questions?