DENIAL OF SERVICE ATTACKS

Leonardo Querzoni
querzoni@diag.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

DENIAL OF SERVICE

What is a Denial of Service attack?

Any attack that reduces or eliminates service availability

Makes the service/data unavailable for authorized users.

Airline hit by 2 DDoS attacks in 8 mths; ticket sales impacted

CYBERATTACK TARGETS DROP IN PORTAL TRAFFIC

DENIAL OF SERVICE

Availability is a core component of the CIA triad

- Its disruption has immediate economic and operational consequences
  - E.g. production stop

It may impact with various degrees of severity

- Complete unavailability
- Severe Performance degradation
- SLA violations
- …

HISTORICAL EVOLUTION OF DOS ATTACKS

1990s

SYN flood, Ping of Death

2000s

Botnets and reflection

2010s

Application-layer attacks

2020s

IOT botnets and protocol abuse

THREAT MODEL FOR DOS ATTACKS

- Attacker goals
  - Disruption, Extortion, Distraction
- Attacker capabilities
  - Bandwidth, Bots, exploits & tech skills
- Victim assets
  - Ingress bandwidth, CPU/memory resources, stateful components
- Network assumptions
  - Spoofing is feasible? Amplifiers are openly available?

ATTACKER MOTIVATIONS

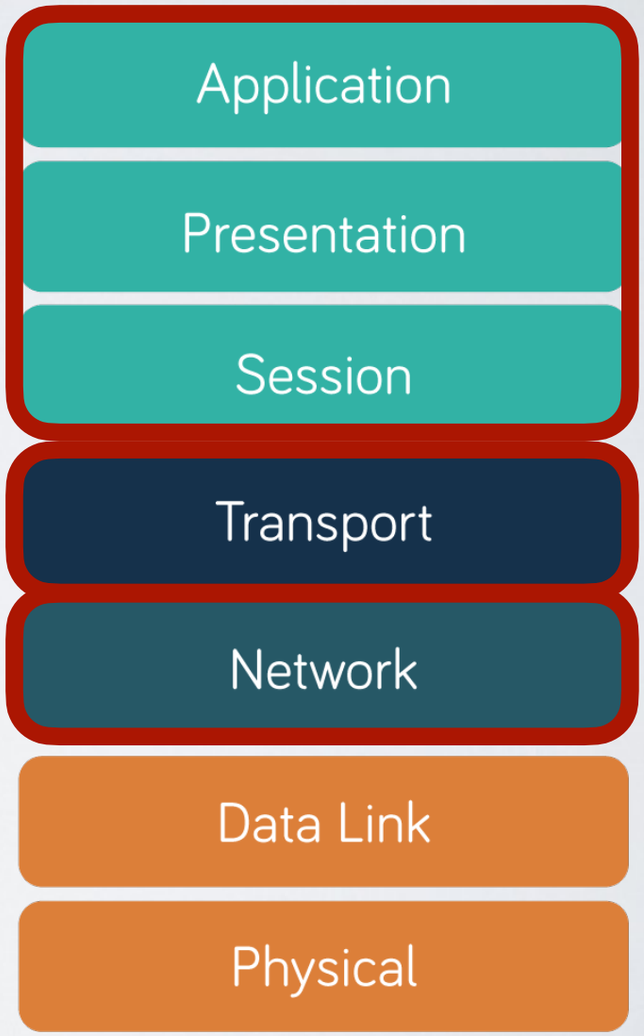
- Financial extortion
  - E.g. DD4BC campaigns (2014–2016). The group “DDoS-for-Bitcoin (DD4BC)” sent ransom notes and launched demonstration floods—then threatened much larger attacks unless victims paid in BTC
- Political/ideological activism (hacktivism)
  - E.g. Operation Payback (Dec 2010). Under the Anonymous banner, hacktivists DDoS’d Mastercard, Visa, and PayPal after the firms cut off donations to WikiLeaks.

ATTACKER MOTIVATIONS

- Competitive sabotage
  - E.g. Spamhaus vs. CyberBunker (Mar 2013). After anti-spam NGO Spamhaus blacklisted ranges associated with Dutch “bulletproof” host CyberBunker, attackers mounted one of the largest DDoS attacks seen to that date.
  - Commercially motivated retaliation to protect spam-tolerant hosting business interests.
- Military and cyberwarfare contexts
  - E.g. Ukraine (Feb 2022, pre-invasion). Coordinated DDoS waves hit Ukrainian government portals (MoD, Armed Forces) and major banks (PrivatBank, Oschadbank), disrupting online services and mobile apps during heightened tensions

DOS ATTACK SURFACE

- Network stack layers
  - L3: saturate bandwidth between client and service (or upstream)
    - E.g. UDP reflection/amplification
  - L4: exhausting state on servers or middleboxes: SYN backlogs, connection tables in firewalls/NATs, or per-flow limits
    - E.g. TCP SYN floods
  - L7: force expensive server work
    - E.g. HTTP GET/POST floods to non-cacheable paths



DOS ATTACK SURFACE

- Application logic
  - Application design choices expand or shrink your attack surface
  - Which endpoints are inherently expensive? Which features fan-out to databases or third-party APIs ? Which inputs can trigger worst-case complexity?
- Shared infrastructure
  - Shared infrastructure we rely upon can be used for reflection/amplification
  - E.g. DNS, Anycast/CDN
- Third-party dependencies
  - Same is true for non-shared but third-party-run infrastructure
  - E.g. External Identity Management systems.

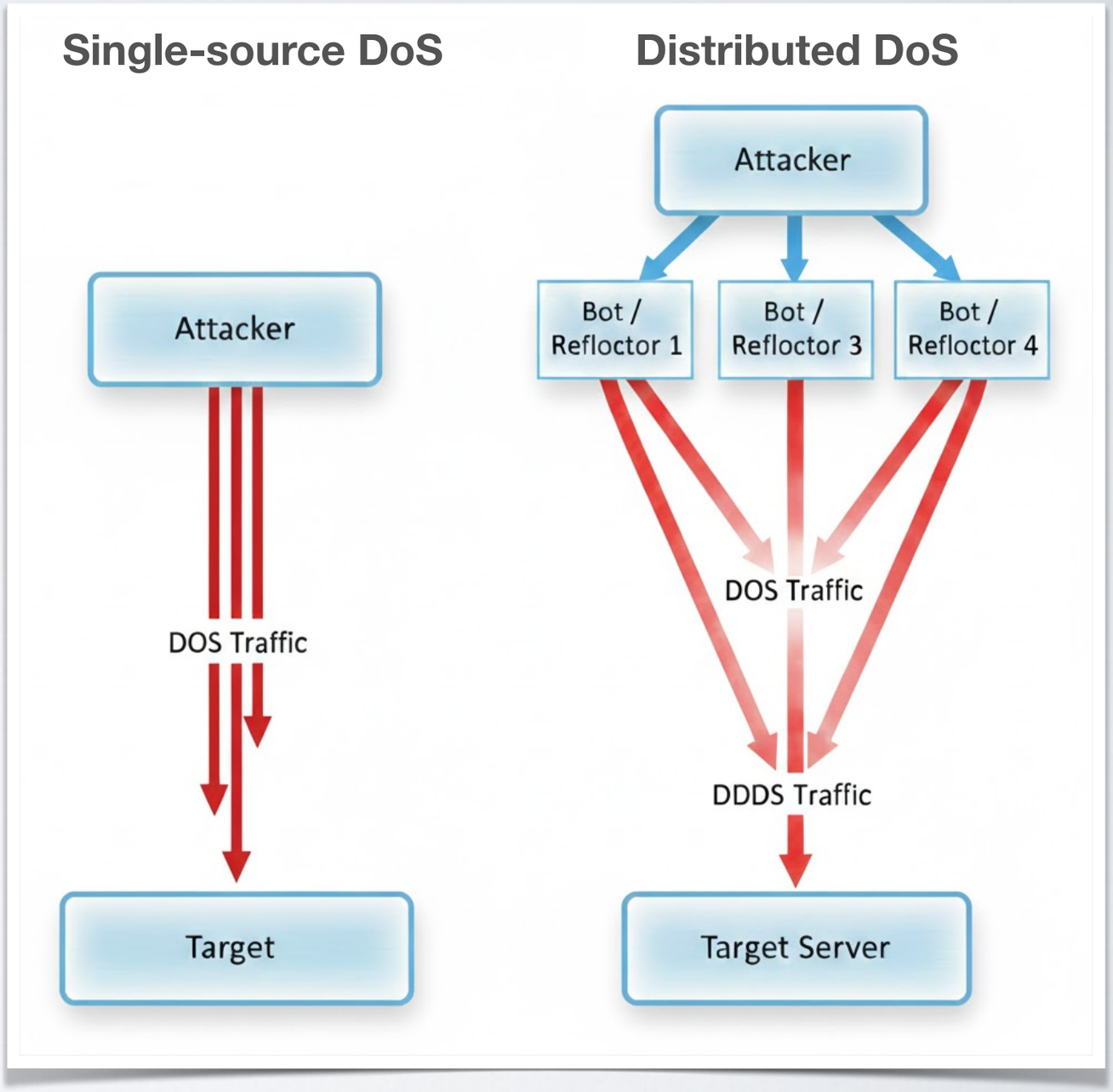
DOS VS DDOS

Attacks may be centralized or distributed:

- Single-source DoS
- Distributed DDoS

Amplification of attack power via distribution

Implications for attribution and mitigation

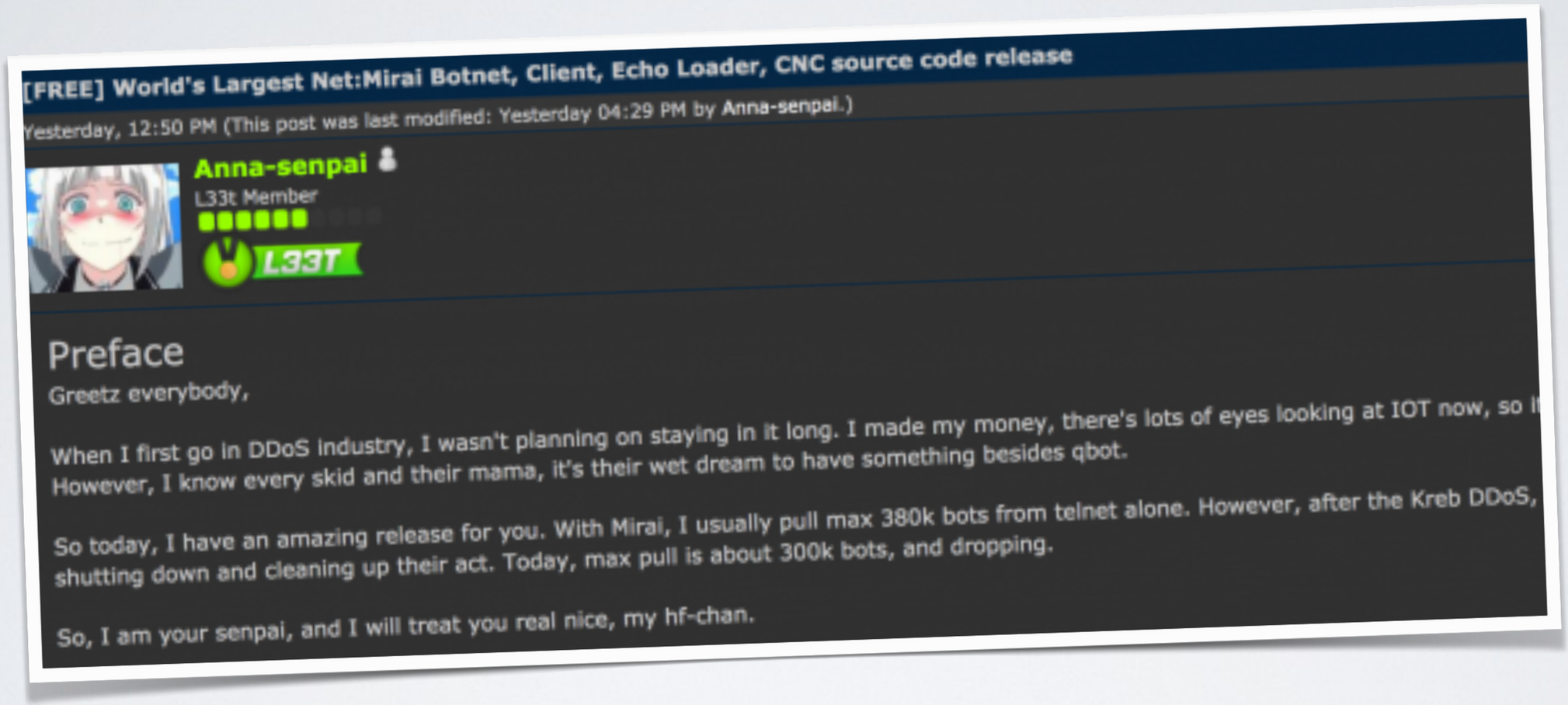


MIRAI

2016 was the year of the first IoT-based botnet: Mirai

MIRAI

Followup: Mirai source code is released for free…



[FREE] World's Largest Net:Mirai Botnet, Client, Echo Loader, CNC source code release

Yesterday, 12:50 PM (This post was last modified: Yesterday 04:29 PM by Anna-senpai.)

Anna-senpai

L33t Member

L33T

Preface

Greetz everybody,

When I first go in DDoS industry, I wasn't planning on staying in it long. I made my money, there's lots of eyes looking at IOT now, so I
However, I know every skid and their mama, it's their wet dream to have something besides qbot.

So today, I have an amazing release for you. With Mirai, I usually pull max 380k bots from telnet alone. However, after the Kreb DDoS,
shutting down and cleaning up their act. Today, max pull is about 300k bots, and dropping.

So, I am your senpai, and I will treat you real nice, my hf-chan.

AISURU

Aisuru is a Mirai-derivative (TurboMIRAI) botnet

- Operated as a DDoS-for-hire service
- Has repeatedly records for both bandwidth and packet-rate in 2025
- Primarily recruits consumer CPE—home/broadband routers, CCTV/DVRs, and similar devices—and generates direct-path floods using medium-size packets and randomized ports/flags
- Q3 2025 record: Cloudflare documents a 29.7 Tbps UDP carpet-bombing burst (approx. 15k destination ports/sec, ~69 secs. Duration) from >500k unique sources.
- Device enrollment exploits supply chain compromise (TotoLink firmware update server) and continuous scanning for new RCE vulnerabilities

CLASSIFICATION OF DOS ATTACKS

DoS attacks are usually categorized as:

- Volumetric attacks
- Protocol attacks
- Application-layer attacks
- Algorithmic complexity attacks

VOLUMETRIC ATTACKS

Volumetric DDoS attacks aim to **saturate bandwidth** and choke the network path to the service

- They remain the most common DDoS category and exploit **cheaply rentable botnets** to generate massive floods
- Measured in Gbps/Tbps
- Attack sizes have **grown exponentially** over the past decade, with recent hyper-volumetric events routinely clearing 1 Tbps, peaking at 29.7 Tbps in 2025 (Aisuru botnet)
- Typical symptoms: high utilization, drops, latency

UDP FLOODS

Network-layer DDoS in which the attacker hammers a target (host or firewall) with **huge numbers of UDP packets** to random or specific ports.

- Because UDP is connectionless, the target must spend work checking for a listening application; if none is present, it typically emits an **ICMP 'Destination Unreachable'**
- UDP is stateless => Packet generation simplicity
- Attacker commonly **spoof source IPs** to stay anonymous

Variants:

- **UDP fragmentation** floods
- **Carpet-bombing** (spray UDP to numerous ports across ranges to evade simple filters)

ICMP FLOODS

ICMP flood overwhelms a target by blasting it with ICMP Echo Request (“ping”) packets at very high packets-per-second (pps).

- The victim spends CPU to parse each request
- Also emits matching Echo Replies, consuming bandwidth in both directions

Attacker may spoof source IP to evade echo replies o use a botnet

Indicators: unjustified spikes in ICMP traffic.

Variants:

- **Smurf**: attacker forges the victim’s IP as the source and sends Echo Requests to a network’s broadcast address. Every host on that network dutifully sends Echo Replies to the spoofed victim, multiplying traffic by the number of hosts

REFLECTION AND AMPLIFICATION

Cyber Threat Actors typically resort to two fundamental techniques to make their DoS attacks more effective:

Reflection

- Attacker sends requests to third-party servers (“reflectors”) while forging the source IP to be the victim’s address.
- Each reflector then replies to the victim, effectively “reflecting” traffic toward the target.
- The attacker stays hidden
- Common on UDP-based attacks where connection state is not maintained

REFLECTION AND AMPLIFICATION

Cyber Threat Actors typically resort to two fundamental techniques to make their DoS attacks more effective:

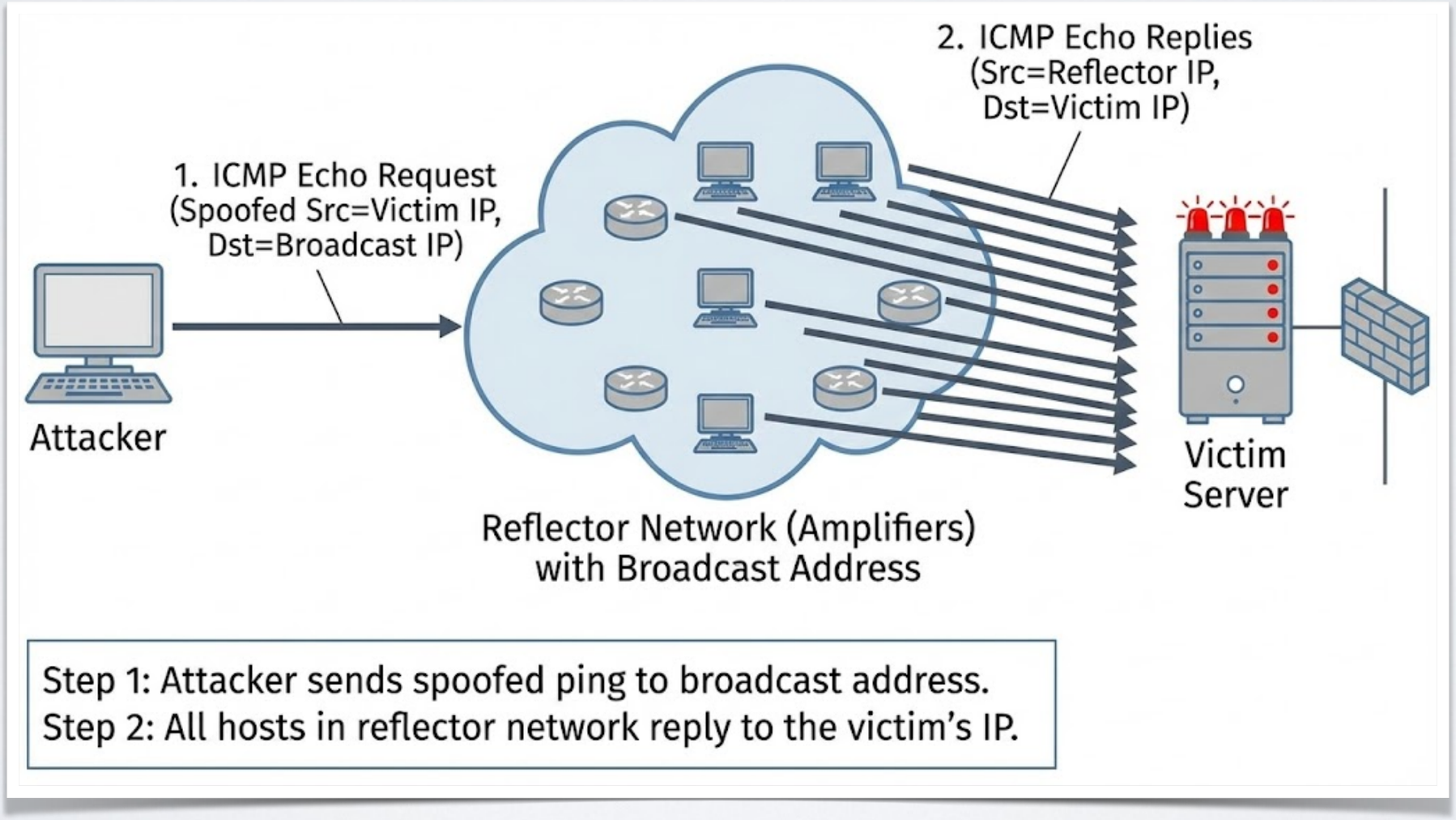
Amplification

- Attacker sends tiny, well-formed UDP queries to Internet-exposed services (e.g., DNS, NTP, SSDP, Memcached).
- Spoofs the source IP to the victim's address.
- Each service then replies to the victim with a larger response.

ICMP FLOODS

Smurf leverages both techniques:

- Uses source IP spoofing for reflection
- Exploits networks open to broadcast for amplification



DNS AMPLIFICATION

The DNS service is a great option for DoS with amplification:

- Is based on UDP
- Replies are often much larger than requests
- Wide availability of open resolvers

Amplification factors > 50x

Variant: the DNS service as a target itself

- **NXDOMAIN flood**: Attackers query non-existent names to force expensive recursion and generate NXDOMAIN replies, bypassing caches and exhausting resource

Case: The Spamhaus 2013 incident is the canonical real-world case of DNS amplification (https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/)

NTP AMPLIFICATION

NTP synchronizes clocks on networked machines and runs over port 123/UDP

- An obscure command, MONLIST, allows a requesting computer to receive information regarding the last 600 connections to the NTP server
- An attacker can spoof the target's IP address and send a monlist command to request that the NTP server send a large amount of information to the target.
- Since the response from the NTP server is larger than the request sent from the attacker, the effect of the attack is amplified.

Amplification close to 200x !

Case: In February 2014, Cloudflare mitigated a DDoS campaign that peaked just under 400Gbps (https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/)

MEMCACHED AMPLIFICATION

Memcached is a high-performance, in-memory key-value cache used to speed up web apps.

- Historically, it could listen on TCP and UDP (default port 11211)
- The attacker locates Internet-reachable memcached servers that still respond over UDP/11211.
- The attacker SETs a large value (e.g., a big blob) into the server
- The attacker then sends a tiny GET request over UDP, but spoofs the source IP to the victim's address.

Amplification can rise up to ~50.000x !!!

Case: in 2018 GitHub was hit by a memcached-based reflection/amplification DDoS that peaked at ~1.35Tbps

MITIGATIONS AGAINST VOLUMETRIC ATTACKS

1) Avoid to expose services toward the public internet if not needed
   - Or impose authentication
   - Or, at least, do not listen on UDP
2) Block UDP spoofing
   - BCP-38 is the best practice for network ingress filtering
   - Can be used by operators to block packets with forged (spoofed) source IP addresses from leaving (or entering) their domains
   - The idea is simple: the ISP bind to to each interface a filter that define the IP space associate to the network reachable through that interface. Incoming packed with IP outside of the allowed ranges are discarded.

PROTOCOL ATTACKS

Exploit protocol state

- Many network/application protocols keep per-connection or per-session state
- Attackers force state creation or expensive processing with minimal traffic
- Impact: memory/CPU exhaustion, connection table overflow,

Target middleboxes or servers

Lower bandwidth, higher impact

- These attacks exploit cost asymmetry: a tiny request can trigger large server-side work.
- Often stealthier than volumetric floods—traffic looks “legit”

TCP SYN FLOOD

Attacker sends a succession of TCP Synchronize (SYN) requests to the target

- Target responds by acknowledging the request and holds the communication open while it waits for the client to acknowledge the open connection.
- In a successful SYN Flood, the client acknowledgment never arrives.

Exhaustion of backlog queues

Countermeasures:

- SYN cookies: defer state allocation; encode minimal state in the SYN-ACK sequence number and only allocate on receiving the final ACK
- Backlog tuning
- Per-client quotas

ACK AND RST FLOODS

A TCP ACK flood is a L4 (transport-layer) DoS where attackers send large volumes of ACK-flagged TCP segments that do not belong to legitimate connections.

- Bypass SYN-centric defenses.
- Force slow-path lookups.
- Egress amplification: Servers receiving stray ACKs may generate RSTs.

A TCP RST flood sends many segments with the RST flag set.

- Force expensive validation on servers/middleboxes.
- When possible, terminates existing connections.

FRAGMENTATION ATTACKS

Fragmentation attacks exploit the way IP (IPv4/IPv6) splits packets into fragments and how endpoints/middleboxes reassemble them.

- **Reassembly cache exhaustion**: flood of first fragments (offset 0, MF=1) with unique IDs per 4-tuple, or mixed fragments that never complete a full set.
- **Tiny-fragment / header-hiding**: split the packet so the first fragment contains only the IP header and a sliver of payload, not the TCP/UDP header.
- **Overlapping fragments**: Two or more fragments that claim overlapping byte ranges with conflicting data. Exploit ambiguity among different devices.

Case: in 2018 a Linux kernel flaw dubbed *FragmentSmack* exposed an inefficient algorithm in kernel's IPv4/IPv6 fragment reassembly path: Impact: CPU saturation.

APPLICATION-LAYER ATTACKS

Goal is to exhaust compute/IO at the origin (web/app servers, APIs, DBs, caches) using legit-looking requests.

- Cost asymmetry: Small requests → big server work
- Common targets: Login/auth, search, report/export, image/PDF render, payment APIs, DNS resolvers.
- Operational symptoms: Rising p95/p99 latency, CPU/thread pool saturation, queue depth growth, 5xx/timeouts, cache miss rate spikes.

HTTP FLOODS

Surge of GET/POST requests that target vulnerable services

- Hit expensive endpoints: search, login/auth, report/export, image/PDF render.
- Bypass caches: random params, unique URLs, no-cache headers for CDNs.
- Protocol features abuse: HTTP/2 multiplexing/rapid reset, aggressive keep-alive, TLS handshake/renegotiation pressure.

Variant: **Keep-alive abuse**

- HTTP/1.1+ keeps a TCP/TLS connection open so a client can send multiple requests without re-handshaking.
- Open many keep-alive connections and leave them idle → consume sockets, memory, and thread/concurrency slots
- Send very slow headers/bodies (slowloris/slow POST) on persistent connections.

SLOWLORIS-STYLE ATTACKS

Is a low-bandwidth HTTP/1.x “slow header” attack

- The attacker opens many connections and sends partial HTTP request headers extremely slowly, just enough to keep each connection alive without ever completing the request.
- Aims at thread/connection pool exhaustion, FD exhaustion, and queue backpressure.
- Differently from other DoS attacks, with slowloris the packet rates are low (thus difficult to track), but the exhaustion of resources is unavoidable.

ALGORITHMIC COMPLEXITY ATTACKS

The attacker crafts legit-looking inputs that force apps into worst-case time/memory paths (e.g., $O(n^2)$, exponential backtracking, collision storms)

- Exploit worst-case performance

Common vectors:

- Regex DoS (ReDoS): patterns with catastrophic backtracking on crafted strings.
- Hash-table collision floods: many POST/JSON keys that collide in maps → insertion becomes $O(n^2)$.
- Parser/decoder stress: deeply nested JSON/XML, extreme recursion.
- Algorithm hotspots: expensive graph/search/report endpoints (e.g., worst-case sort/merge/aggregate).

DOS AGAINST CRYPTOGRAPHIC OPERATIONS

Abuse of expensive crypto paths (handshakes, signature/KDF checks, certificate validation) to exhaust CPU/memory on gateways and origins

- TLS handshake floods: force repeated ECDHE + certificate chain validation; historically renegotiation abuse (now mostly disabled).
- Client-cert / signature storms: many requests that need RSA/ECDSA/EdDSA verification (e.g., mTLS, webhook/JWT checks).
- Password hashing pressure: mass login attempts that trigger bcrypt/scrypt/Argon2 checks (high work factors).
- PKI validation hits: push OCSP/CRL lookups or oversized/pathological X.509.

MITIGATIONS

1) Absorb upstream, not at your origin

- Use a scrubbing provider/CDN with Anycast and large edge capacity.
- Advertise/route your prefixes or put your apps behind managed L3/L7 DDoS protection so they handle volumetric floods before traffic reaches you.

2) Hide and harden the origin

- Serve via CDN/WAF; block direct-to-origin (allowlist only your edge POPs).
- Cache aggressively (static + API cache where safe), enable stale-while-revalidate to keep serving under stress.

3) Rate limit and prioritize at the edge

- Per-client quotas, token buckets/leaky buckets per endpoint.
- Challenge/attestation for suspicious clients.

MITIGATIONS

4) Protocol-aware L3/L4 controls

- SYN cookies and SYN proxy; drop NEW-without-SYN; limit RST/ACK anomalies; enable uRPF/BCP-38 with your ISP to reduce spoofing.
- For fragments: reassemble at the edge, drop overlaps.

5) Application-layer fail-fast

- Enforce strict limits (header/body size, regex timeouts, request time budgets).
- Authenticate/authorize before heavy work; paginate and cap result sizes; queue and defer expensive tasks.

6) Crypto-cost controls

- TLS 1.3, session resumption (tickets/PSK), OCSP stapling.
- Cache verified tokens/keys; verify cheap things first.

MITIGATIONS

7) HTTP/2–HTTP/3 hygiene

- Cap concurrent streams, header table size, and RST/stream-creation rates; quickly drop misbehaving connections.
- Guard against HTTP/2 rapid reset-style patterns.

8) Resource isolation & autoscaling

- Stateless services, per-service concurrency caps.
- Autoscale frontends and separately scale backends; apply circuit breakers and graceful shedding (Retry-After, serve stale).

9) Observability + early detection

- Dashboards/alerts for SYN backlog