Denial of Service

System Security (CM0625, CM0631) 2023-24
Università Ca’ Foscari Venezia

Riccardo Focardi
www.unive.it/data/persone/5590470
secgroup.dais.unive.it

Università Ca’Foscari Venezia

Introduction

Denial of Service (DoS)

Increasingly **popular** attack that compromises the **availability** of a service

Example: service is "**flooded**" by many ***spurious requests*** that make it impossible to respond to valid requests

Introduction

Distributed Denial of Service (DDoS)

DoS is particularly effective when launched from **many devices**. Increasing strength in the years ...

- **~400 Mbps** in 2002
- **~100 Gbps** in 2010
- **~300 Gbps** Spamhaus, in 2013
- **~600 Gbps** BBC, in 2015

⇒ easily exceed the bandwidth!
But usually **not very long**
(~30min, *botnets-for-hire*)

Introduction

Distributed Denial of Service (DDoS and IoT)

In 2016 a **new kind of attack** on Dyn, a DNS provider

- long, many **hours**
- involved multiple attacks from over **100,000 devices**
- **IoT** (Internet of Things) devices, such as **webcams** and **baby monitors**
- reached a **~1.2 TBps** peak

Definition

[NIST SP 800-61]

DoS: An attack that **prevents** or **impairs** the **authorized use** of networks, systems, or applications by **exhausting resources**

Targets:

- network bandwidth
- system resources
- application resources

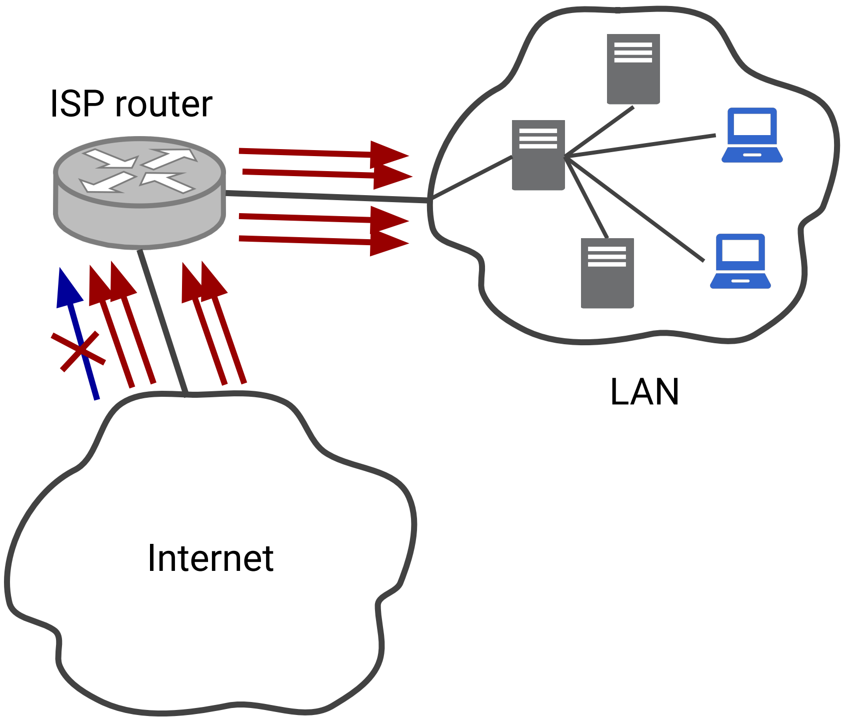
Target: network bandwidth

Network bandwidth: capacity of links connecting a server to the Internet

⇒ Usually the link to the Internet Service Provider (ISP)

If incoming traffic exceeds the bandwidth **packets will be discarded**

⇒ **Legitimate packets discarded** if malicious ones exceeds network bandwidth



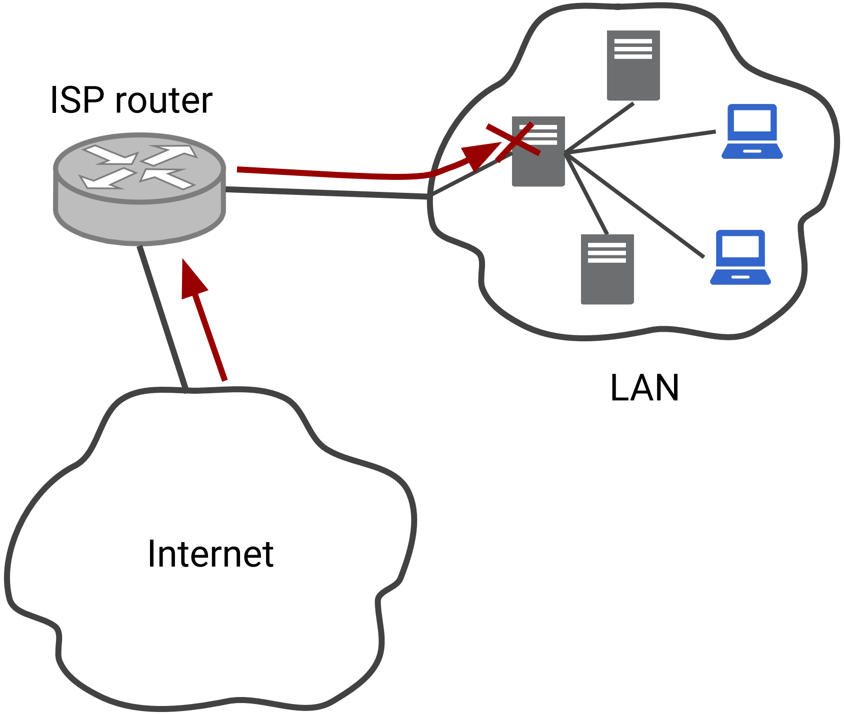
Target: system resources

Network handling resources: required to implement network protocols (e.g. buffers)

⇒ When limit is reached new network connections are refused
Example: TCP connections

"***Poison packets***" might trigger bugs that break network services

Examples: *ping of death, teardrop*



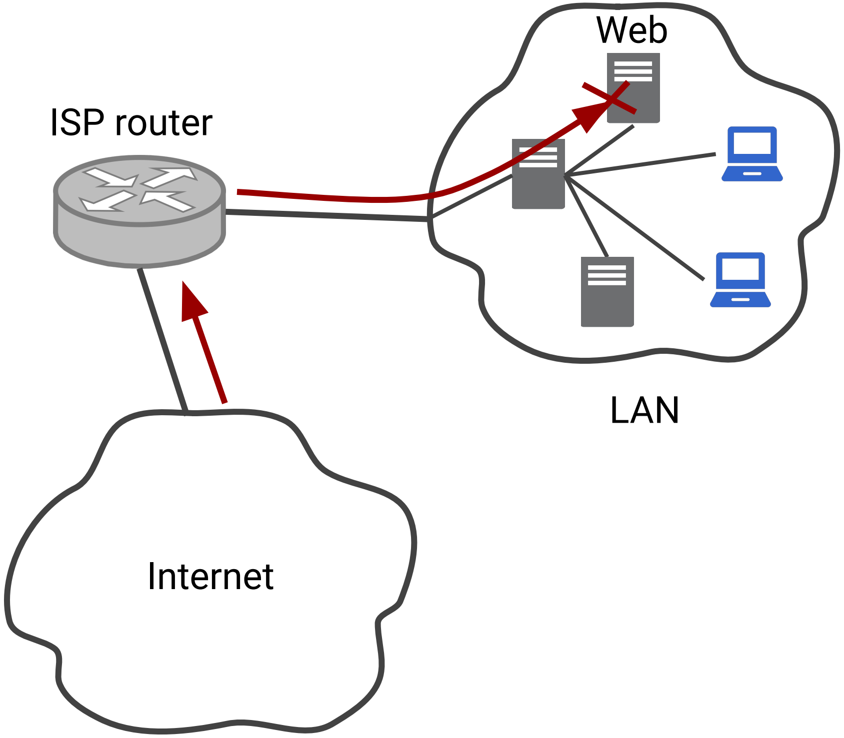
Target: application resources

Application resources: required to accomplish tasks

⇒ When limit is reached application becomes unresponsive
Example: Excessively complex queries to a database

Destructive attacks, exploiting bugs, that crash the application

Examples: *Piggybacked SQLi*



Target

- **Network bandwidth**
- System resources
- Application resources

Flooding attacks

Flooding attack: overwhelm network capacity

ICMP flooding: the ***Internet Control Message Protocol*** (ICMP) is used to send error messages and operational information

Example: **`ping`** allows for testing connectivity (`-f` option **floods the server**)

UDP flooding: attacker targets a UDP service (es. diagnostic echo)

TCP flooding: attacker targets TCP services

Target server might either **respond**, generate a ICMP **destination unreachable** or **reject** the packet

⇒ Responses increase the load!

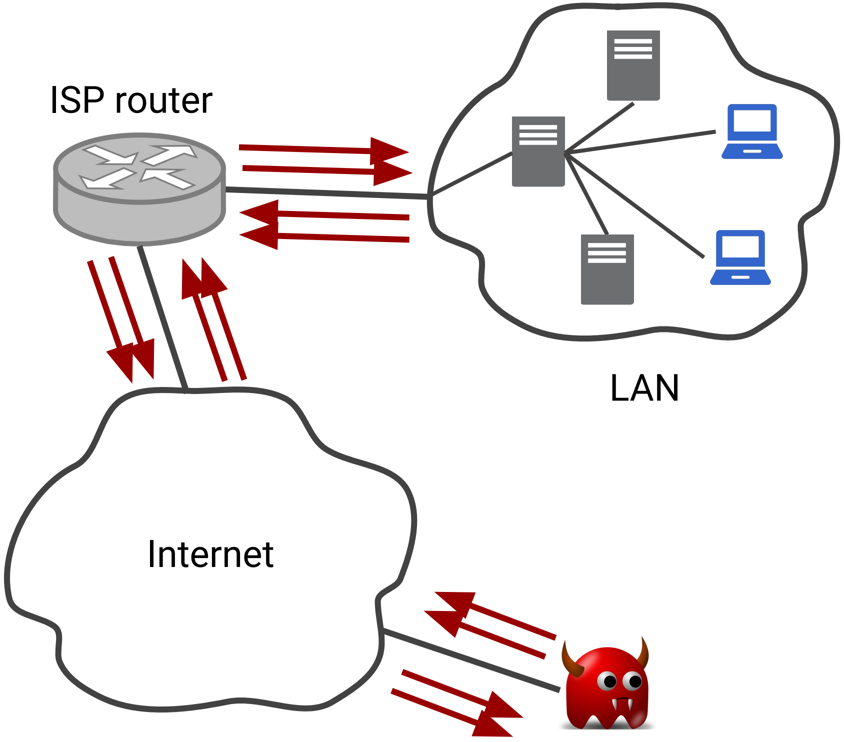
Simple flooding

Simple flooding attack: overwhelm network capacity **from a single host**

Example: ping with -f option

However:

- Source **easily identified** (legal actions taken)
- Target will respond “**reflecting**” the attack back



Source address spoofing

Source address spoofing: attacker use *raw socket interface* to change source address

Randomly selected source addresses

⇒ Responses will be **scattered** around the Internet

⇒ Possible errors packets from spoofed address will go towards the target and **contribute to DoS**

Source address spoofing makes it **hard to identify** the attacker

Cause: TCP/IP does not ensure that source address really corresponds to the originating host

It would be necessary to (manually) query the logs of **traversed routers** in order to identify the trajectory

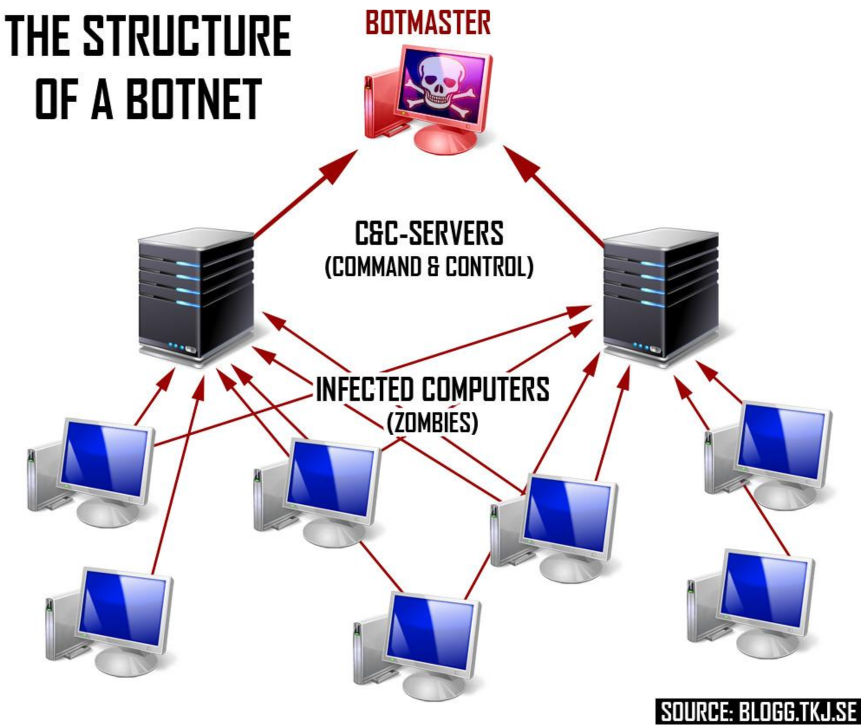
Distributed DoS (DDos)

Botnets: a collection of **zombie devices** under the attacker control

Botnets <u>are hired</u> for DDos

- **~40%** of DDoS in 2015 were from **botnets for hire**

Flooding coming from thousands of hosts easily reaches **Gbps** bandwidth



Target

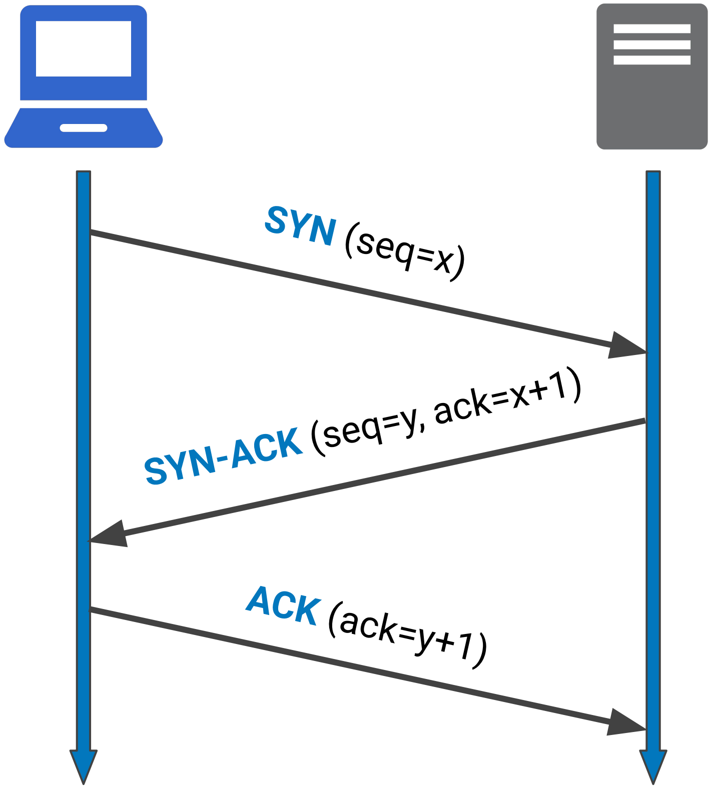
- Network bandwidth
- **System resources**
- Application resources

SYN spoofing

This attack **overflows** the tables used to manage TCP connections

TCP uses a **three-way handshake** to establish a connection:

- IP lost packets are transparently **resent**
- Applications using TCP won't notice lost packets and retransmissions

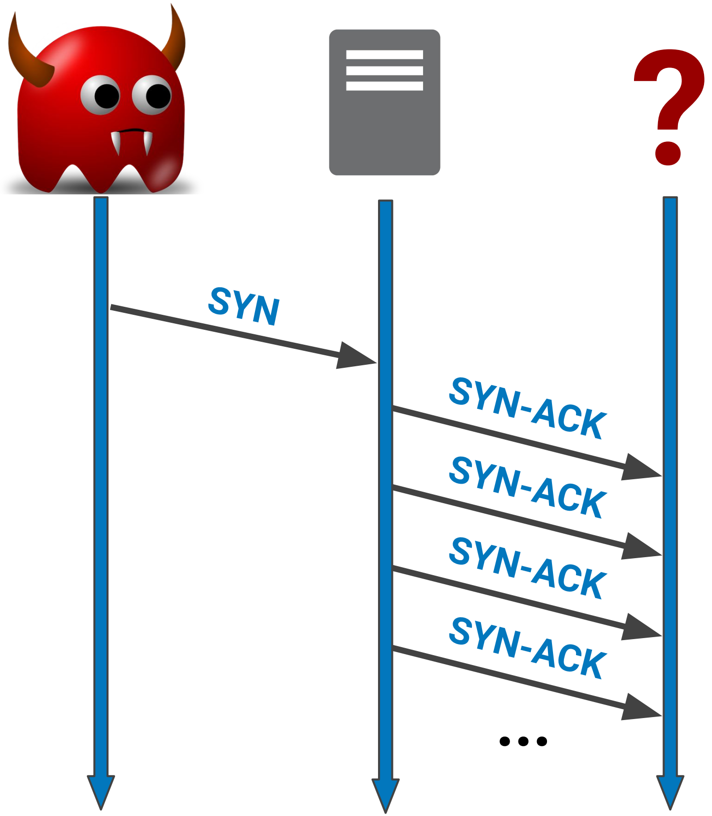


SYN spoofing (ctd)

Attack scheme:

1. Attacker sends **SYN packets** with **spoofed source** addresses
2. For each **spoofed source S**:
   a. Server sends **SYN-ACK** to S
   b. If Server timeouts and N_S<MAX
      $N_S = N_S+1$
      goto a
   c. Delete connection with S

⇒ Table of TCP requests **overflows**



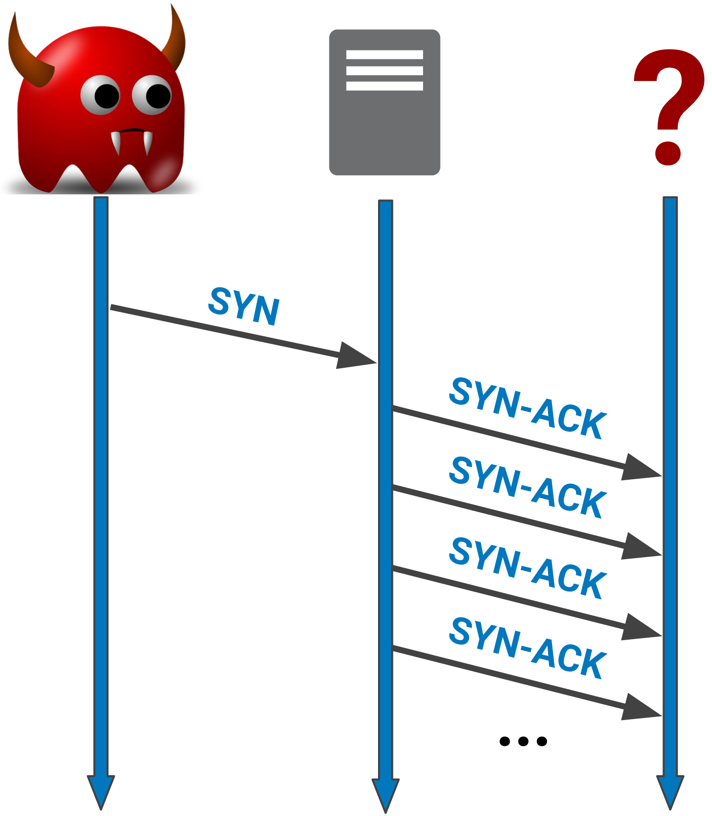
SYN spoofing (ctd)

Attacker sends **enough forged requests** to keep the table full

⇒ Server is **cut off** from the Internet

NOTE: Using **random** spoofed addresses make the probability of not getting a **RST** (reset) answer high

The volume of **SYN** requests is **low** and far from link capacity



Target

- Network bandwidth
- System resources
- **Application resources**

Application protocol flooding

Attacker **floods an application protocol**

Examples:

- ***Session Initiation Protocol*** (***SIP***) used in VoIP. INVITE requests go through proxies and consume system/network resources
- **HTTP requests** can be heavy (e.g. download of large file)

Slowloris: a particular DoS attack that leverages server multi-threading

- start many HTTP requests **without completing** them
- keep the **connection alive** by sending new lines, periodically

Consumes **all available web server connections** (in terms of internal system/application resources)

DoS techniques

- Reflection
- Amplification

Reflection attacks

Attacker sends packets to an ***intermediary*** with a spoofed source address of the ***target***

The ***intermediary*** responds to the actual ***target***

- If response is **larger** than request, attack is also amplified
- **Tracing** is hard if attacker uses many intermediaries

Examples: **DNS**, **SNMP** and **ISAKMP** has been exploited for reflection (they can generate **large** responses)

TCP SYN reflection: Attacker can send **SYN** so that intermediary sends **SYN-ACK** which in turns generates a **RST** packet

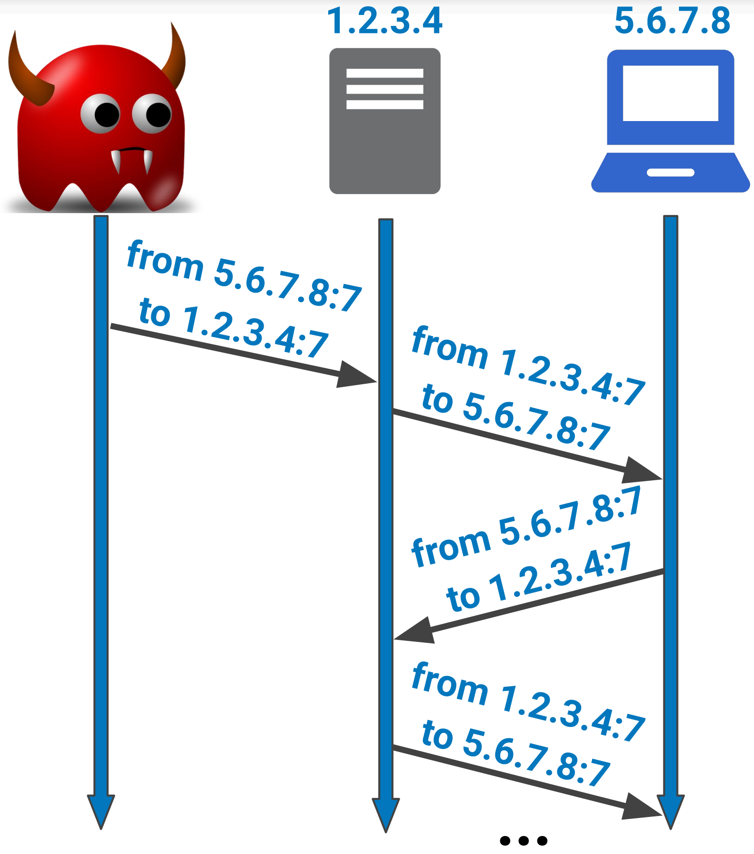
⇒ Both **SYN-ACK** and **RST** flood target's network

Reflection “loop”

When echo service (port 7) is enabled **reflection loops** are possible

Example: The attacker sends a packet to **1.2.3.4** port **7**, with spoofed source address **5.6.7.8** port **7**

- Intermediary **echoes** to target
- Target **echoes** to Intermediary
- … (loop)

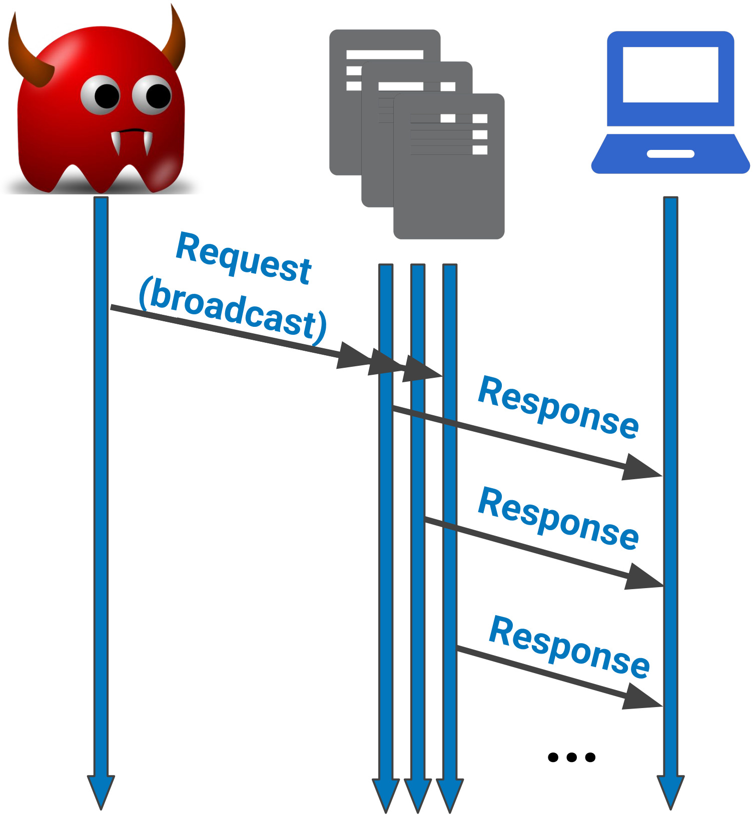


Amplification attacks

Amplification: generating multiple response packets with a single request

Example: send a packet to the **broadcast** address of a network with spoofed address

⇒ **all hosts** (with the service enabled) **will respond** to the target



Defenses

DoS cannot be fully prevented: attackers that can flood a service with **legitimate requests** will limit traffic from other users

DoS can be “incidental”: important **news** make legitimate users overload referenced web sites

Defenses

what and when

Prevention and mitigation
(**before** the attack)

Detection
(**during** the attack)

Source traceback
(**during** and **after** the attack)

Reaction
(**after** the attack)

Prevention: spoofed source addresses

Solution 1: **filtering** spoofed source address as close as possible to the originating host

Example: where the organization's network **connects** to the Internet

Filtering spoofed source addresses is a standard security recommendation (RFC 2827) which is **too often disregarded**!

Solution 2: ensure that the **path back** to the claimed source address is the one being used by the current packet

Example: CISCO implemented this however when **routing is asymmetrical** (path A $\longrightarrow$ B and B $\longleftarrow$ A differ) this solution is too strict

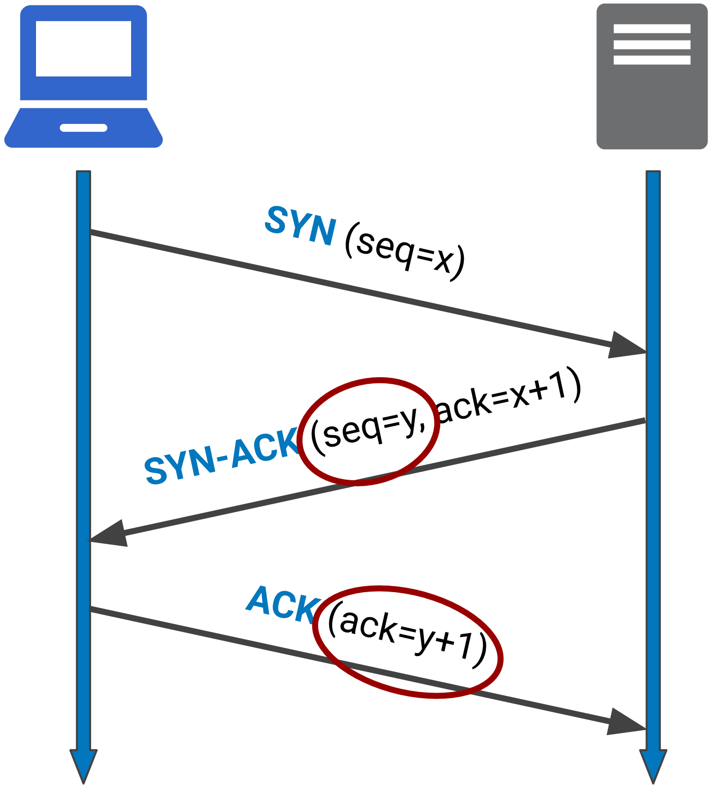
Multihoming: Connecting to many networks for reliability/performance

Prevention: SYN spoofing attack

Make the protocol "stateless" by encoding state information directly in the **SYN-ACK** sequence number y
⇒ No TCP request **table overflow**!

When the **ACK** y+1 is received the server can reconstruct state information from y

Example: "SYN Cookies" in FreeBSD and Linux (similar idea in Windows)



Mitigation: rate limits and random drops

ICMP and UDP flooding to diagnostic services can be mitigated by imposing **limits on packet rates**

Similarly, SYN spoofing attacks can be mitigated by limiting the **connection rate** to a certain service

Table overflow of SYN spoofing can be mitigated by **randomly dropping** connections

IDEA: overflow is a probable sign of attack, randomly dropping a connection will more likely **drop an attacker's connection**

Could drop a legitimate connection but it is **better than full DoS**

Other prevention techniques

Block **broadcast** (**amplification**)

Block/limit **suspicious services** and combinations of ports (**reflection**)

Check **human interaction**, e.g. with captcha (**application** resources)

Keep systems **up-to-date and secured** (do not become part of a **botnet**)

Monitor systems, especially high-performance, well-connected servers (potential **intermediary**)

Use **mirrored** and **replicated** servers to increase reliability and **resilience** to DoS attacks

Detection, source traceback and reaction

Detection: capturing packet flows and analyzing them. If the attack is identified

- suitable **filters** can be activated
- **bugs** can be fixed
- **alternate backup** servers can be activated
- ...

Source traceback: necessary for **legal actions**, need collaborating ISPs (can be complex)

Reaction: analyzing the attack and response in order to gain benefit from the experience and to improve future handling. The organization's security can be **improved** as a result