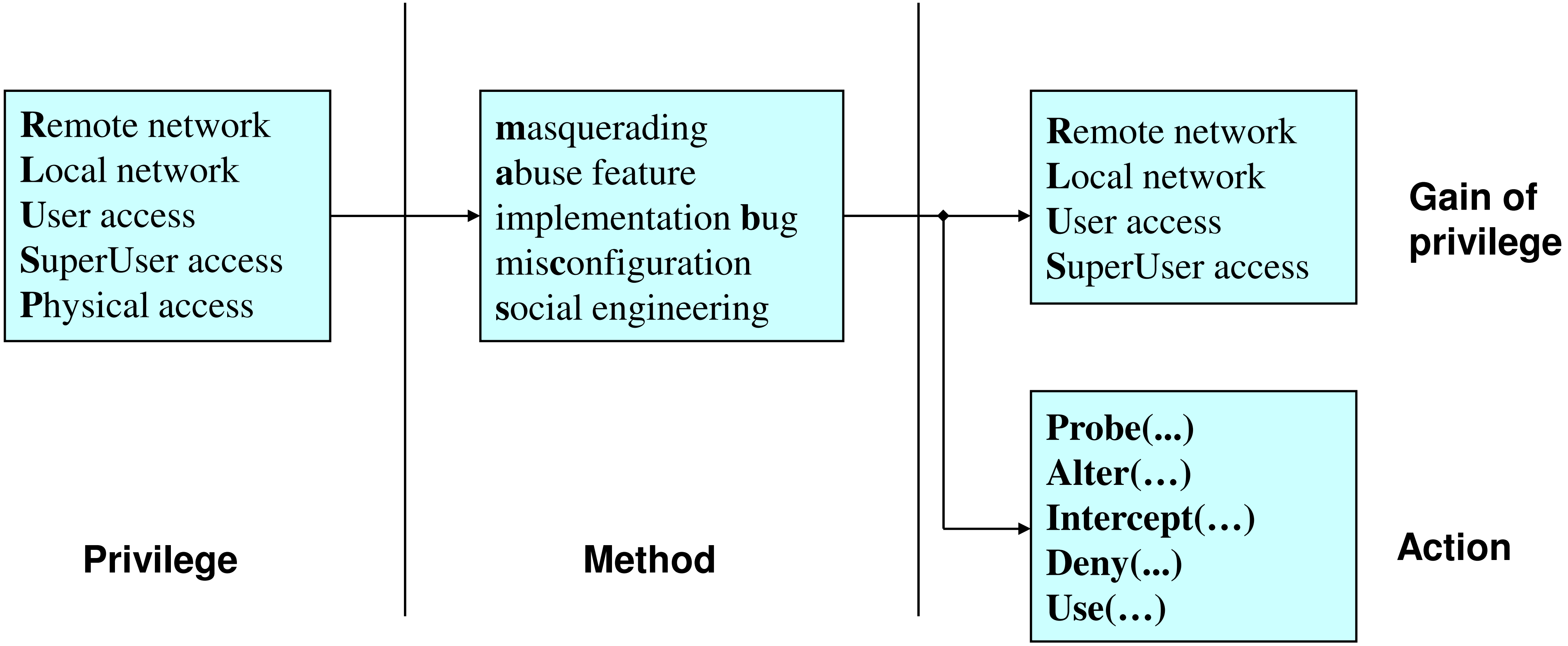
Attack Classifications

Jacques CAZIN

Summary



Remote network
Local network
User access
SuperUser access
Physical access

masquerading
abuse feature
implementation **bug**
mis**c**onfiguration
social engineering

Remote network
Local network
User access
SuperUser access

Probe(...)
Alter(…)
Intercept(…)
Deny(...)
Use(…)

Privilege

Method

Gain of privilege

Action

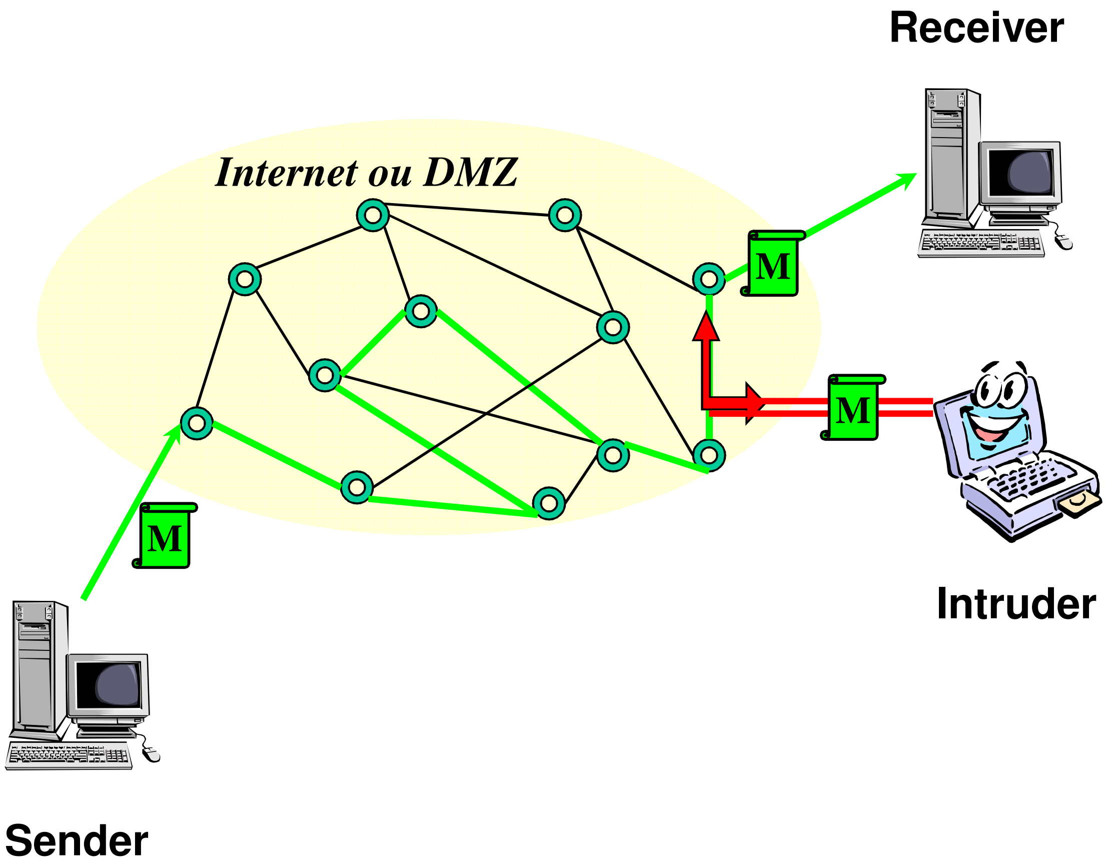
3. Attack Examples

Main attack “classes”

- **Sniffing**
  - ⇨ Probe
- **Spoofing**
  - ⇨ Masquerading
- **Flooding**
  - ⇨ Deny of service
- **Scanning**
  - ⇨ Probe(services)
- **Hijacking**
  - ⇨ Intercept
- **Virus and Trojan Horse**

Packet sniffing (L-a-Probe or R-a-Probe)

- **Principles**
  - ⇨ Listening or Intercepting packets transmitted through a local network or through internet to collect “interesting” information:
    - User id, Password (not always encrypted…)
    - Smart card, credit card numbers
    - Type and version of devices
    - ...



Spoofing

- **Principle:**
  - ⇨ Masquerading: forging packets with false address to cheat the identity of a given machine

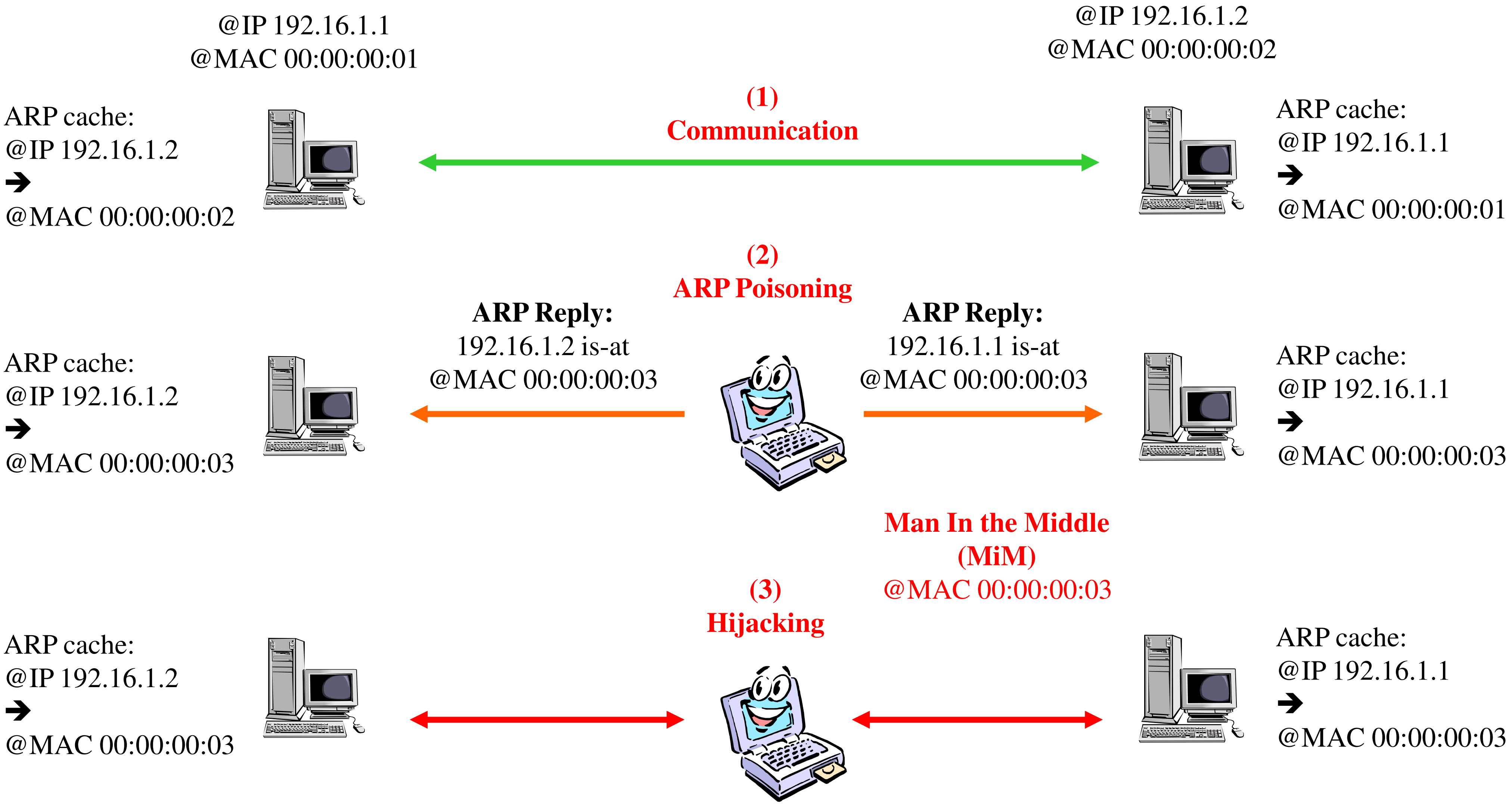
- **Most common spoofing:**
  - ⇨ ARP Spoofing (also called ARP poisoning)
  - ⇨ ICMP Spoofing
  - ⇨ UDP Spoofing
  - ⇨ TCP Spoofing

ARP Poisoning

- **Principle of ARP protocol (unconnected protocol):**
  - ⇨ In the ARP protocol, each “request” is broadcast to the other machines of a given LAN
  - ⇨ Each machine keeps in its cache the correspondence @IP/@MAC
  - ⇨ The cache is updated when the machine receives an “ARP reply” (even though it did not send an “ARP request”)

- **Principle of the attack:**
  - ⇨ The intruder sends “ARP reply” messages with @IP that does not correspond to @MAC
  - ⇨ Applications:
    - Deny of service
    - Hijacking

ARP Hijacking (R-m-Intercept)



@IP 192.16.1.1
@MAC 00:00:00:01

@IP 192.16.1.2
@MAC 00:00:00:02

ARP cache:
@IP 192.16.1.2
➔
@MAC 00:00:00:02

(1)
Communication

ARP cache:
@IP 192.16.1.1
➔
@MAC 00:00:00:01

(2)
ARP Poisoning

ARP cache:
@IP 192.16.1.2
➔
@MAC 00:00:00:03

ARP Reply:
192.16.1.2 is-at
@MAC 00:00:00:03

ARP Reply:
192.16.1.1 is-at
@MAC 00:00:00:03

ARP cache:
@IP 192.16.1.1
➔
@MAC 00:00:00:03

Man In the Middle
(MiM)
@MAC 00:00:00:03

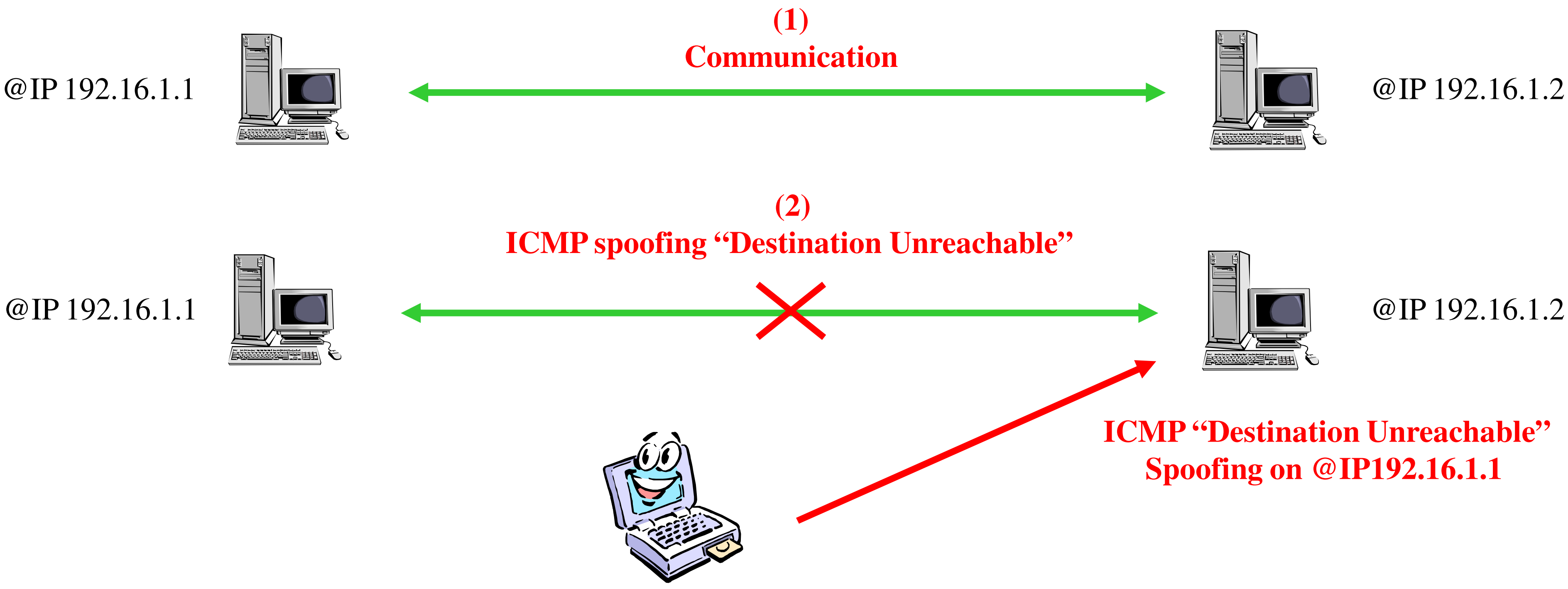
(3)
Hijacking

ARP cache:
@IP 192.16.1.2
➔
@MAC 00:00:00:03

ARP cache:
@IP 192.16.1.1
➔
@MAC 00:00:00:03

ICMP Spoofing

- **Examples of ICMP spoofing:**
  - ⇨ With ICMP packet “Redirect” ➔ Man in the Middle attack (Hijacking)
  - ⇨ With ICMP packet “Echo Request” ➔ Smurfing (see section on DOS attacks)
  - ⇨ With ICMP packet “Destination Unreachable” ➔ To close a connection

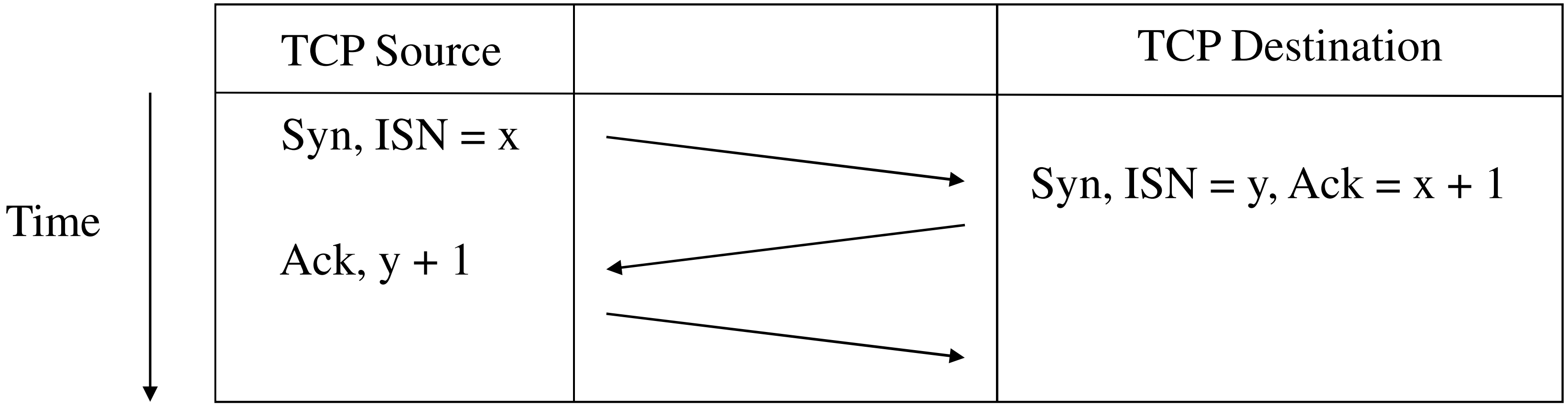

(1)
Communication
@IP 192.16.1.1
@IP 192.16.1.2
(2)
ICMP spoofing “Destination Unreachable”
@IP 192.16.1.1
@IP 192.16.1.2
ICMP “Destination Unreachable”
Spoofing on @IP192.16.1.1

UDP Spoofing

- **Attack simple to make**
  - ⇨ Unconnected protocol
- **Possible applications:**
  - ⇨ DoS attack
    - Example: see the Fraggle attack below
  - ⇨ Hijacking attack
    - Example: Hijacking on the “talk” service (social engineering attack)

TCP Spoofing

- **Much more complex**
  - ⇨ Connected protocol
  - ⇨ Use an ISN
    - Initial Sequence Number
  - ⇨ The ISN is then incremented each time new data are acknowledged



Time

TCP Source

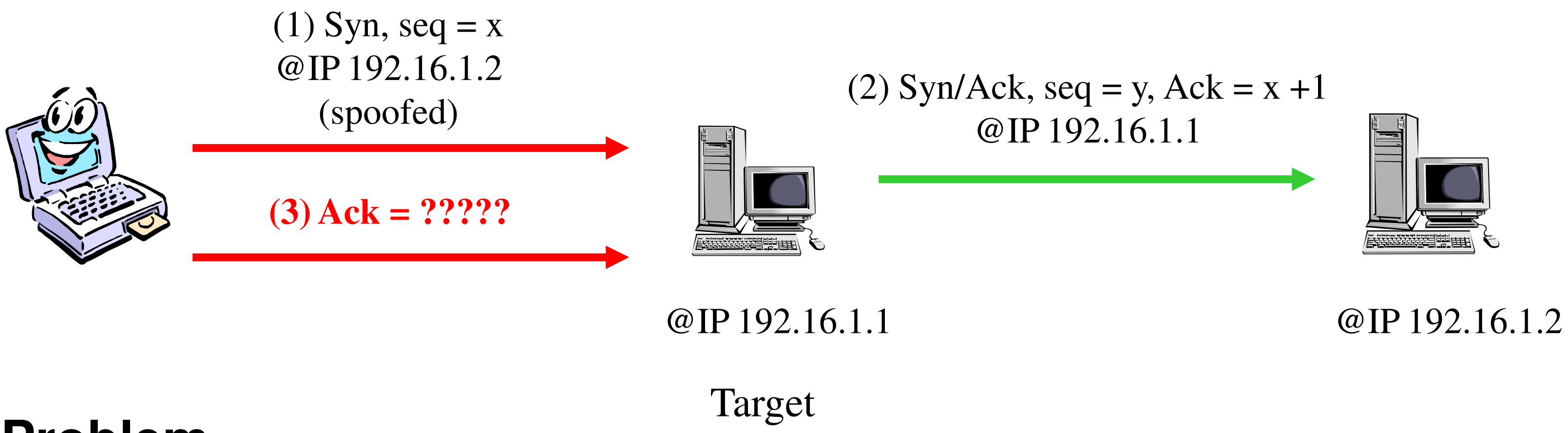
TCP Destination

Syn, ISN = x

Syn, ISN = y, Ack = x + 1

Ack, y + 1

TCP Spoofing (2)



- **Problem**
  - ⇨ What is the ISN sent by the target ?
- **Solution**
  - ⇨ Possibility to forecast the ISN
  - ⇨ The difficulty depends on the OS
  - ⇨ Quite easy on Windows (1 ≤ ISN ≤ 50)
- **Application of TCP spoofing**
  - ⇨ Rlogin on the target (sometimes the system does not ask a password and relies on the IP address of the source)

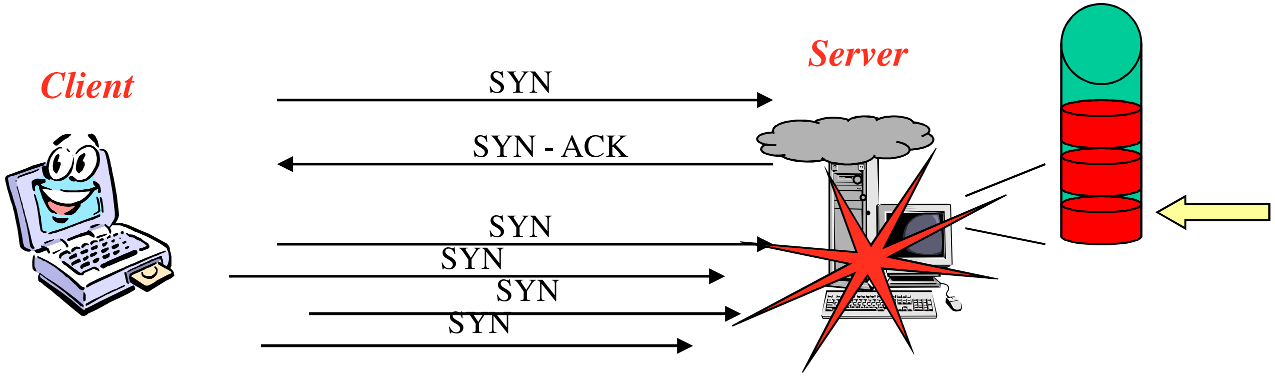
Flooding

- **Principle:**
  - ⇨ Sending a large number of messages so that the receiver cannot handle all of them
  - ⇨ Leads to a Deny of Service or a Distributed Deny of Service (DDOS)

- **Most common flooding**
  - ⇨ TCP flooding (or Syn flooding)
  - ⇨ UDP flooding
  - ⇨ Smurfing (example of ICMP flooding with packets “Echo Request”)

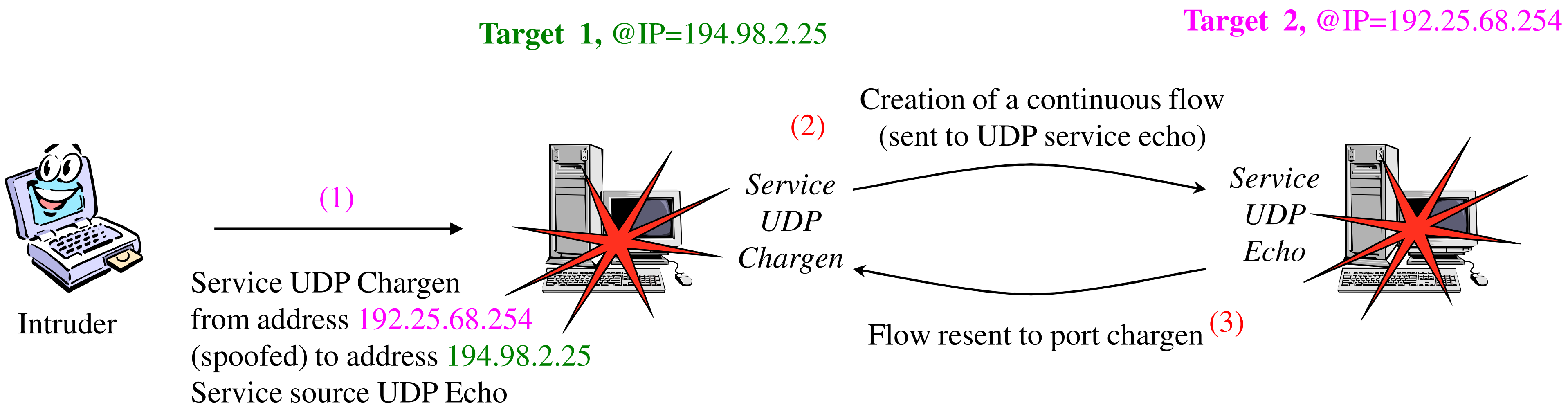
SYN flooding (R-a-deny(temporary or administrative))

- **Principle:**
  - ⇨ Three steps to create a TCP connection:
    - “Syn”, “Syn Ack” and “Ack”
  - ⇨ “Half open” connection:
    - When a “Syn” is sent but there is no “Ack” sent when message “Syn Ack” is received
  - ⇨ Each “half open” connection are stored in the stack
  - ⇨ Leads to a deny of service when too many “Half open connection” are open
  - ⇨ **Administrative Deny of Service** when the attack leads to a stack overflow
  - ⇨ **Temporary Deny of Service** when there is timer to cancel too long Half Open connection so that the stack does not overflow



UDP flooding: Fraggle attack (R-a-deny(temporary))

- **Principle:**
  - ⇨ Use two UDP services:
    - “chargen” (port 19): to create a continuous flow
    - “echo” (port 7): each received packet are resent to the source
  - ⇨ Enables the intruder to make a deny of service on two machines

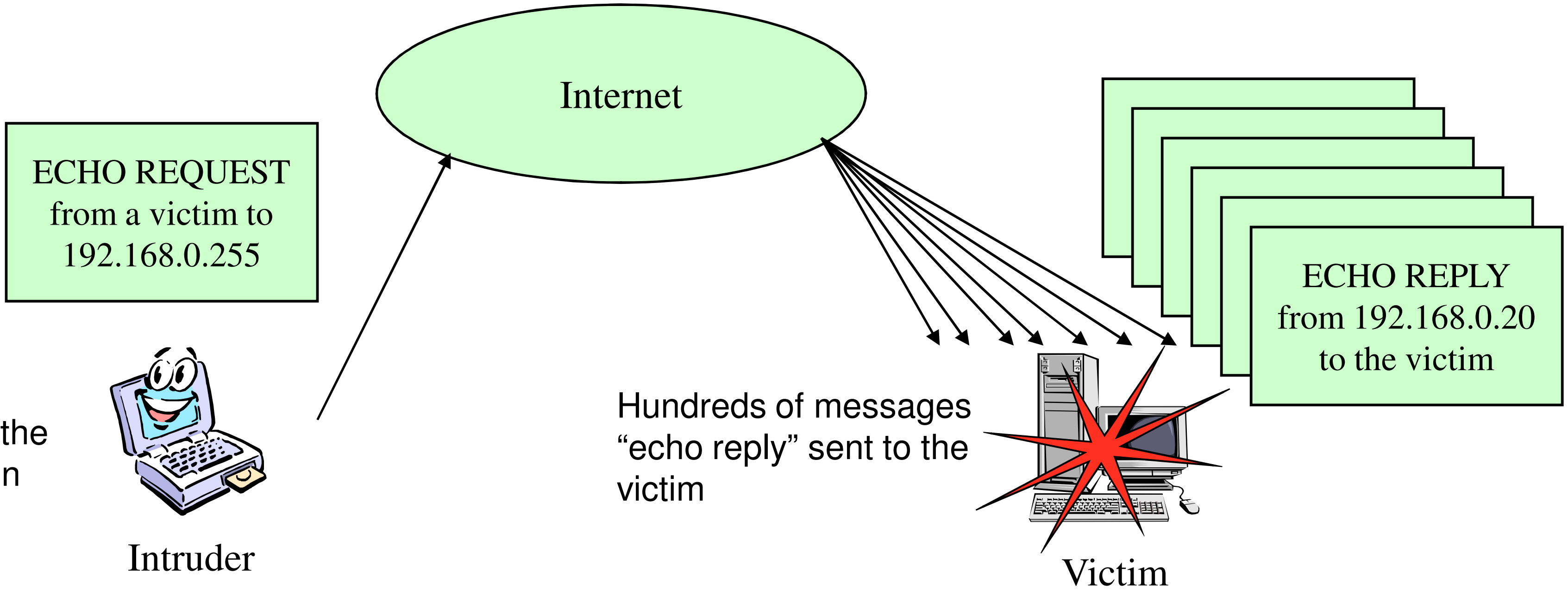

Target 1, @IP=194.98.2.25
Target 2, @IP=192.25.68.254
Intruder
(1)
Service UDP Chargen
from address 192.25.68.254
(spoofed) to address 194.98.2.25
Service source UDP Echo
(2)
Service
UDP
Chargen
Creation of a continuous flow
(sent to UDP service echo)
Service
UDP
Echo
Flow resent to port chargen
(3)

ICMP flooding: Smurfing (R-a-deny(temporary))

Use the address « broadcast »
(corresponding to xxx.xxx.xxx.255)

If a machine sends a message “echo request” at the address “broadcast”, each machine of the corresponding local network sends a message « echo reply »

Sends a message “echo request” with the IP address of a given victim (spoofing)



Internet

ECHO REQUEST
from a victim to
192.168.0.255

Intruder

ECHO REPLY
from 192.168.0.20
to the victim

Hundreds of messages “echo reply” sent to the victim

Victim

Effect of smurfing: amplifies the effect of flooding
Up to 255 messages received by the victim for one message sent by the intruder

Scanning: examples

- **General objective:**
  - ⇨ Obtaining a list of open ports of a given system
- **TCP SYN scanning (half-open scanning)**
  - ⇨ Sending a SYN message, waiting for a SYN-ACK and then RESET
- **TCP FIN scanning**
  - ⇨ Sending a FIN message and waiting for a RESET (closed port) else open port
- **UDP ICMP port unreachable scanning**
  - ⇨ To scan UDP service
  - ⇨ Sending a packet and waiting for a message “ICMP_PORT_UNREACH” (open port)
    Else closed port

  - ⇨ See section “Vulnerability assessment scanners” for further details

Virus and Trojan Horses

- **There are thousand of virus and Trojan Horses**
- **Example: Back Orifice 2000 (bo2k)**
  - ⇨ Creation of a back door
  - ⇨ To take control of a given system

Back Orifice 2000

- **Step 1:**
  - ⇨ Encapsulating bo2k in to an “attractive” file so that the victim will install bo2k into his system
- **Step 2:**
  - ⇨ Connection between the intruder and bo2k on a given port (example: 8080)
- **Step 3:**
  - ⇨ Take control of the victim
    - Start or stop services
    - Modify or download files
    - Etc.

Other examples

- Winnuke
- Land Attack
- Christmas Tree Attack
- Ping of death

WinNuke (R-b-Deny(Administrative))

- **Principle:**
  - ⇨ The attack works on windows95 and windows NT
  - ⇨ Sending a packet with “URGENT” flag set to 1
    - On port 139 (NetBios)
    - But also on other ports used by Windows
  - ⇨ The “URGENT” flag specifies that there are “urgent” data in the packet
  - ⇨ The attack works if there is no “normal” data after the “urgent” data
    - Leads to a deny of service (blue screen)

Other examples of DOS attacks

- **Land Attack (R-b-Deny(Administrative))**
  - ⇨ Principle:
    - Sending a packet with the IP source address equal to the IP target address
- **Christmas tree attack (R-b-Deny(Administrative))**
  - ⇨ Principle:
    - Sending a packet with all the TCP flags set to 1
- **Ping of death (R-b-Deny(Administrative))**
  - ⇨ Principle:
    - Sending packet longer than 65 535 bytes
- **Etc.**

Example of an attack to get a root access

- **ShellCode (R-b-S)**
  - ⇨ Performs a Buffer Overflow
  - ⇨ Example: Red Code

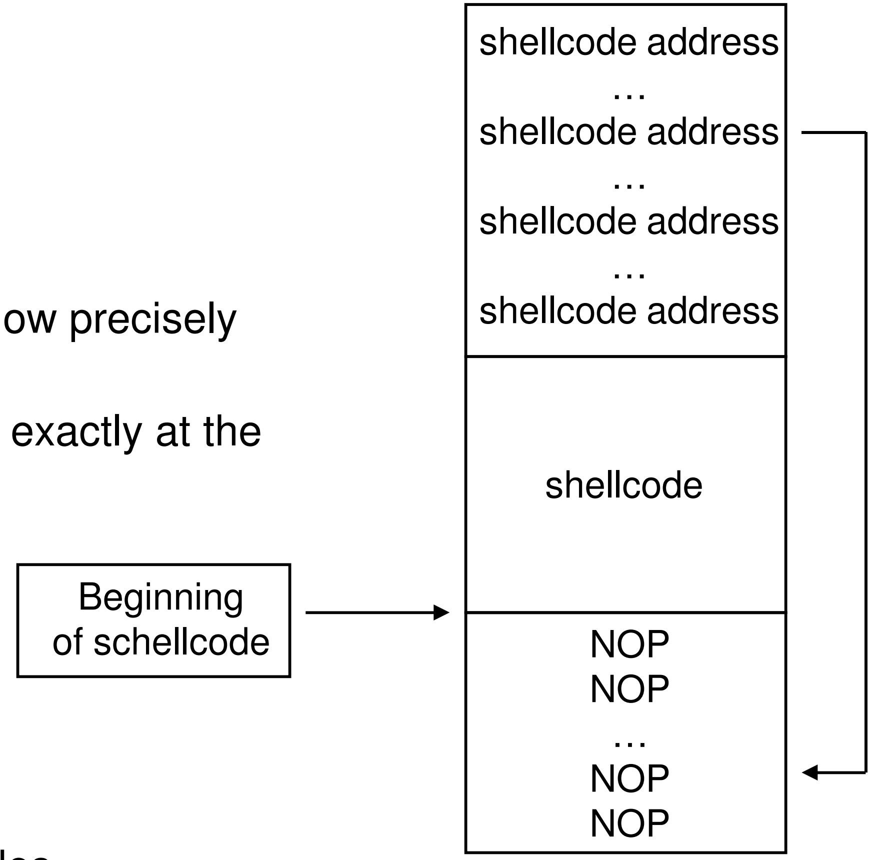
- **Principle :**
  - ⇨ Bad management of dynamic memory
  - ⇨ No separation between the program code and data stored in the stack
  - ⇨ The volume of inserted data is larger than the allocated memory size
    - During its execution, a sub-routine overwrites the return address
    - Enables the execution of a shellcode
  - ⇨ Classical vulnerabilities exploited:
    - Functions on characters string (sprintf)

ShellCode

- **Stack management**
  - ⇨ Function call
    - Start:
      - – Context saving
      - – Static domain creation
    - Program execution
    - End:
      - – Context restoration
- **Problem:**
  - ⇨ The return address might be overwritten

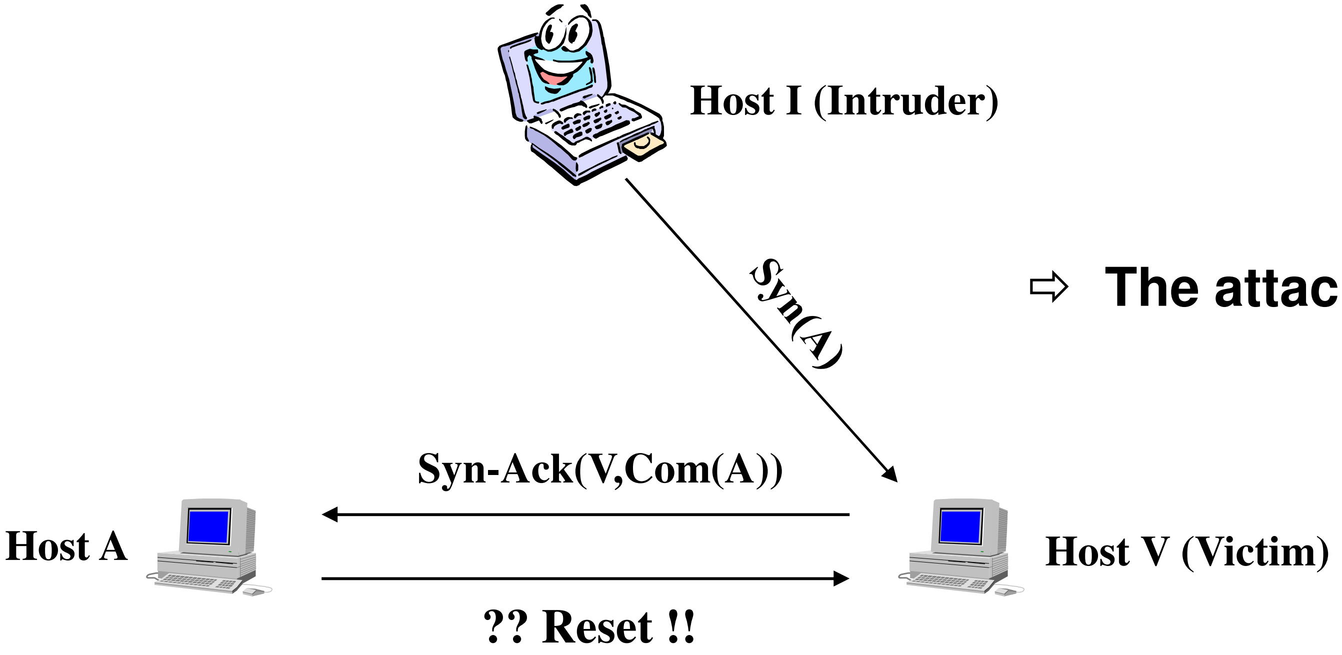
ShellCode shema

- **Three parts:**
  - ⇨ NOP
    - Padding
    - Because, the intruder does not know precisely the shellcode location
    - Problem to set the return address exactly at the beginning of shellcode
  - ⇨ Shellcode main instructions
  - ⇨ Shellcode address



- **Comment:**
  - ⇨ The NOP part is easily detected
  - ⇨ But, existence of polymorphic shellcodes
    - Use libraries of equivalent instructions

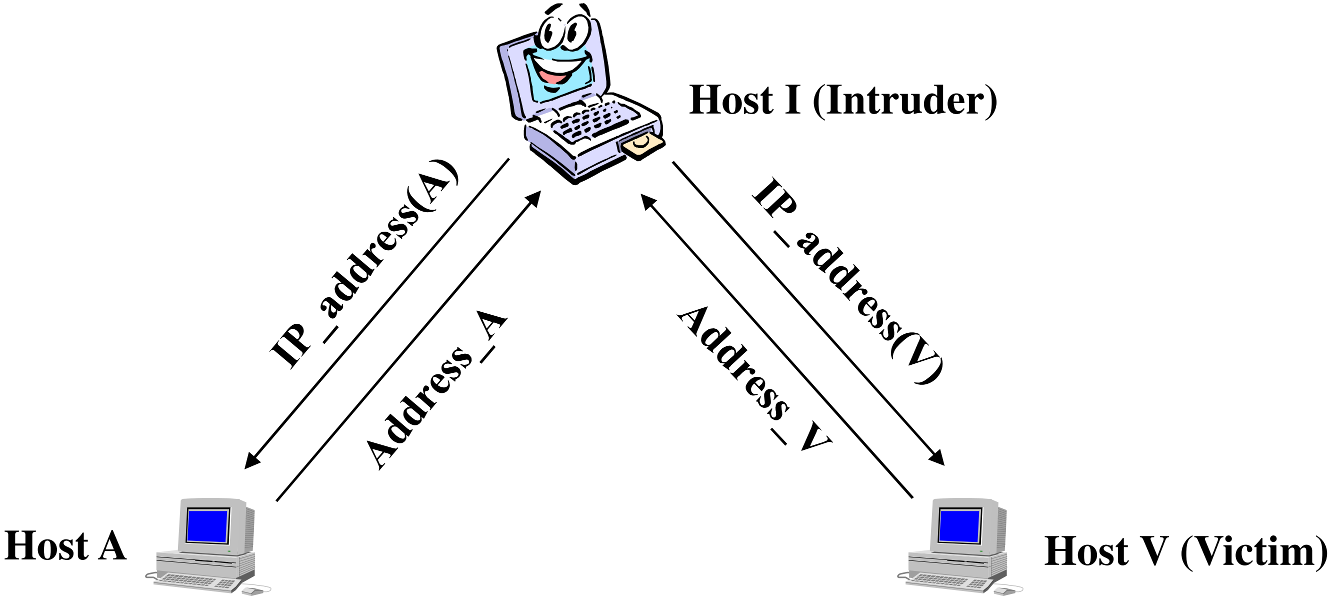
Example of an attack scenario: the Mitnick Attack



⇨ **The attack fails**

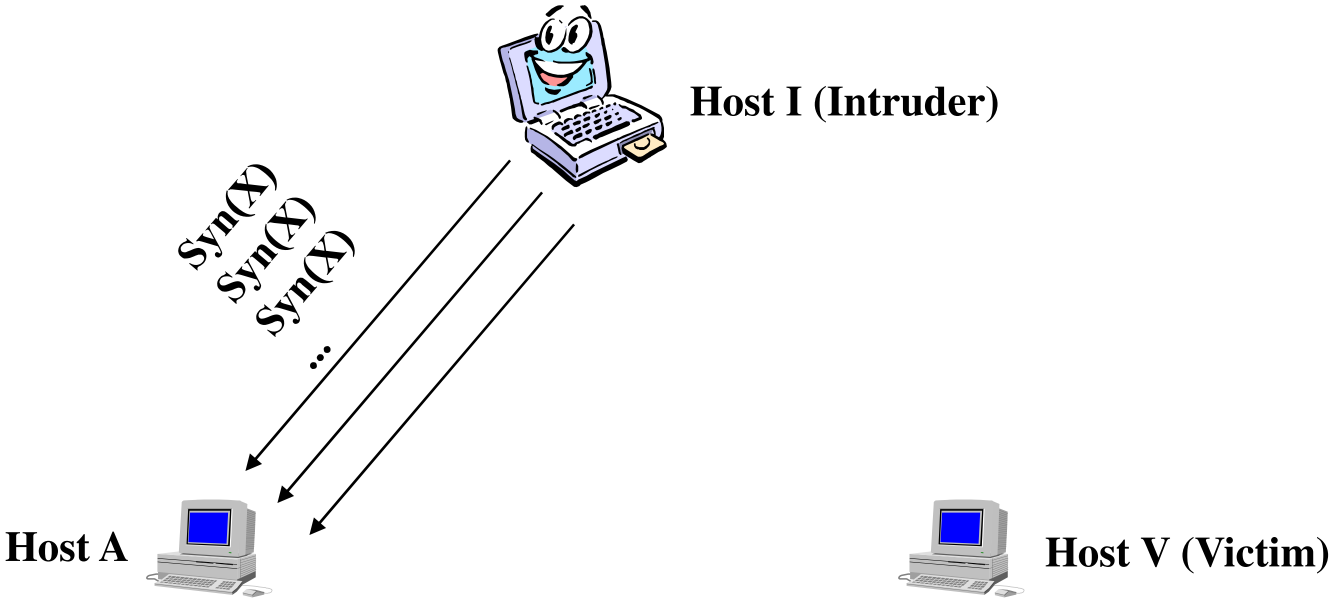
The Mitnick Attack

- **Step 1 : sniffing the IP address of A and V**



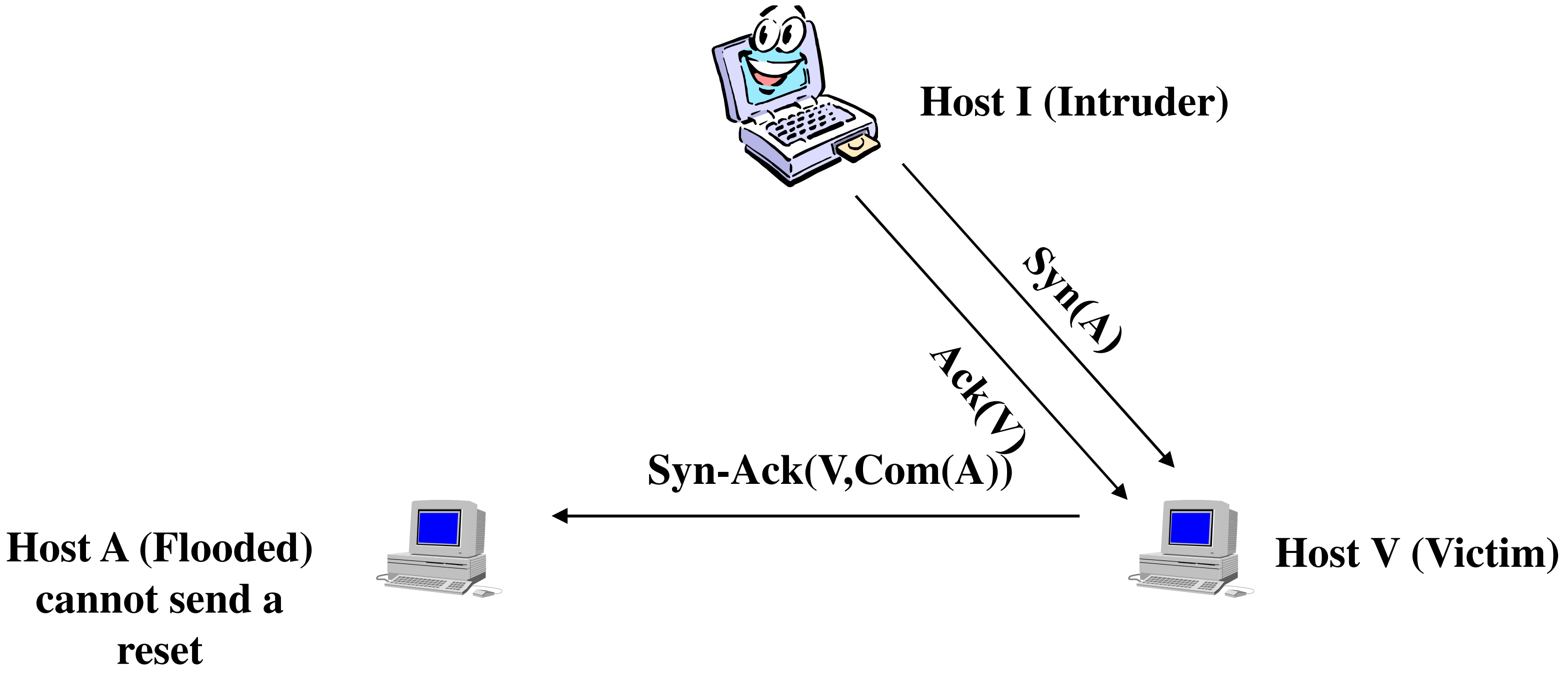
The Mitnick Attack

- **Step 2: SYN flooding of A**

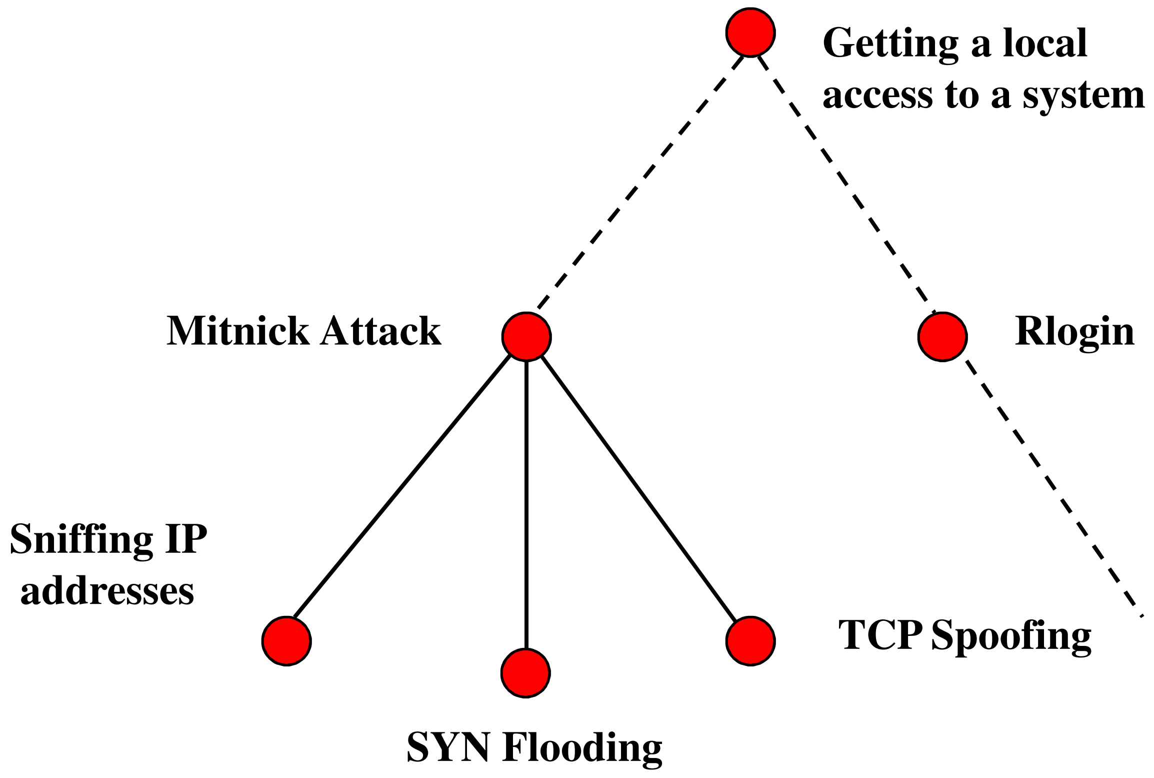


The Mitnick Attack (3)

- **Step 3: TCP spoofing of V**



Attack scenario based on the Mitnick Attack


Getting a local access to a system
Mitnick Attack
Rlogin
Sniffing IP addresses
TCP Spoofing
SYN Flooding