

# Course 3: Network Security, Section 4

Pascal Meunier, Ph.D., M.Sc., CISSP  
updated November 29, 2005

Developed thanks to the support of Symantec Corporation,  
NSF SFS Capacity Building Program (Award Number 0113725)  
and the Purdue e-Enterprise Center

Copyright (2004) Purdue Research Foundation. All rights reserved.



## Course 3 Learning Plan

- Architecture
- Physical and link layer
- Network layer
- **Transport layer**
- Application layer: DNS, RPC, NFS
- Application layer: Routing
- Wireless networks
- More secure protocols: DNSSEC, IPSEC, IPv6

## Learning objectives

- Understand the main differences between UDP and TCP
- Learn how UDP can serve as an exploitation vector
- Learn how TCP is vulnerable

# Transport Layer Vulnerabilities

- Transport layer protocols
  - UDP
    - ❖ Best effort delivery
      - Letter in the mail, hope it gets there (and does most of the time)
        - » Connectionless
  - TCP
    - ❖ Reliable
    - ❖ Establishes connections and monitors deliveries
      - Similar to packages requiring signatures at delivery

# Ports

- Ports dynamically address ("bind") IP packets to a process
  - Socket data structures keep the mapping information
    - ❖ Need to "bind" a socket (port) to a process
- Ports range from 0 to 65535
- Ports 0-1023 are reserved for well-known services
  - Require root (in UNIX) access to listen on those ports
- UDP and TCP ports
  - Usually the same port number is assigned to a service for both UDP and TCP (if the service can use both)
  - This is called "multiplexing" in RFCs
    - ❖ A protocol can interact with many processes

## UDP Issues

- All lower layer issues, with similar attacks
  - IP spoofing
  - IP and link layer broadcast (amplification)
  - IP fragmentation
  - ARP spoofing
  - Link layer
- New possibilities
  - Network services and applications can be contacted and attacked with UDP packets that exploit the lower level issues
  - Traffic amplifying applications

# UDP Amplifier Attack

- Fraggle
  - Broadcast UDP packet sent to the "echo" service
  - All computers reply (amplification)
  - Source IP was spoofed, victim is overwhelmed

## Questions

- Which ICMP attack resembles the Fraggle attack?
- Which other services could be used instead of the "echo" service?

## Answers

- Smurf
- All the "small services"
  - daytime
  - chargen (RFC 864)
  - quote of the day (RFC 865)
  - ...
- Quote of the day may be enabled by people thinking it's cool or cute

# Personal Amplification Attacks

- Car as a weapon (physical amplification)
  - Gun
- "For a good time call zzz-zzzz" in public lavatories
- Someone subscribes you to many mailing lists
- When your personal information is sold and you get many telemarketer calls (before do-not-call lists) and junk mail after
  - Registering a product
  - Getting a credit card
  - Buying something online

## UDP Ping-Pong

- Chargen service replies with a UDP packet to any incoming packet
- Spoof a packet from host A's chargen service to host B's chargen service
  - Computers keep replying to each other as fast as they can
- Variants use the echo service on one of the hosts
  - Or even the same host (CVE-1999-0103)
    - ❖ a.k.a. UDP bomb, UDP packet storm

# UDP Ping-Pong

- Any service or application that issues a UDP reply no matter what is the input packet (e.g., error message) is vulnerable
  - daytime (port 13)
  - time (port 37)
- Do you know of another UDP service that answers no matter what?

## Example Hosts Vulnerable to UDP Ping-Pong

- Routers and firewalls!
- Cisco IOS 11.x had chargen and echo enabled by default
  - Date
- Other services
  - Quote of the day (RFC 865)
  - Active Users (RFC 866)
  - Daytime (RFC 867)
  - UDP Kerberos v5 (port 464)
  - Any service that responds (e.g. with an error message) to any packet

# Amplification Using UDP Packets

- Key: Applications that reply with large packets to small requests
  - e.g., games
    - ❖ BattleField 1942
    - ❖ Quake 1 (CAN-1999-1066)
    - ❖ Unreal Tournament
- Hosts can be attacked by using these applications as amplifiers, with forged source IP packets

# Exploits Through UDP

- Resource Exhaustion
  - Windows 98 and Windows 2000 Java clients allow remote attackers to cause a denial of service via a Java applet that opens a large number of UDP sockets, which prevents the host from establishing any additional UDP connections, and possibly causes a crash.
    - ❖ CAN-2001-0324
- Sniffing and Spoofing
  - NAI Sniffer Agent allows remote attackers to gain privileges on the agent by sniffing the initial UDP authentication packets and spoofing commands.
    - ❖ CAN-2000-1159

# Exploits Through UDP

- Exploitation of other flaws (anonymous)
  - Interactions between the CIFS Browser Protocol and NetBIOS as implemented in Microsoft Windows 95, 98, NT, and 2000 allow remote attackers to modify dynamic NetBIOS name cache entries via a spoofed Browse Frame Request in a unicast or UDP broadcast datagram.
    - ❖ CAN-2000-1079
- Traffic amplifiers
  - DNS allows remote attackers to use DNS name servers as traffic amplifiers via a UDP DNS query with a spoofed source address, which produces more traffic to the victim than was sent by the attacker.
    - ❖ CVE-1999-1379

# Exploits Through UDP

- Self-connection
  - Quake 2 server allows remote attackers to cause a denial of service via a spoofed UDP packet with a source address of 127.0.0.1, which causes the server to attempt to connect to itself.
    - ❖ CAN-1999-1230
    - ❖ Similar to UDP bomb

## Discussion and Conclusions

- UDP does not in itself introduce new vulnerabilities, but makes the exploitation of IP layer vulnerabilities easy.
  - Makes applications more difficult to design to prevent amplification and ping-pong effects
- When is UDP needed?
  - DNS
    - ❖ Normal hosts query DNS servers using UDP in practice
      - UDP also used for other DNS functions (more on this later)
  - Streaming video, Voice-over-IP
- Is your LAN used to attack a third party via UDP?
  - Did some computers in your LAN get compromised?

# TCP

- Establishing connections
  - SYN flood attack
- Is this packet relevant?
  - Initial sequence number predictability
  - RST attacks
- TCP Scanning
  - tcptraceroute

# TCP Vocabulary

- TCP is “Stateless” (ambiguous word)
- Refers to the network, not the hosts!
  - Phone conversations require a network path to be established
  - TCP doesn’t change network paths
  - Each packet is independent of others
- Clients and servers maintain states, which makes them vulnerable to resource exhaustion attacks

# TCP Flags

- TCP packets have one-bit flags
- Flags are used to specify the meaning of the packet.
  - SYN (Start of connection)
  - ACK (Acknowledge)
  - FIN ("FINish" or French for "end")
  - RESET
  - PUSH
  - URGENT

# TCP Connections

- To establish a connection:
- Client sends packet with SYN flag (“Hello!”)
- Server responds with packet with both SYN and ACK flags (“Hello, how are you?”)
- Client responds with packet with ACK flag (“Fine, thanks”).
- This establishes that packets can be sent both ways and provides the proof to both hosts.

## TCP SYN Scans

- If someone sends you a SYN packet for a port that's closed, you are supposed to respond with a packet with RST and ACK flags (“I got your message but I don't want to talk to you”).
- Sending SYN packets to find out which ports are open on which machines is known as port scanning
  - Source IP address may be spoofed to hide the true source
    - ❖ Only 1 in x is from the real source

## TCP ACK Scans

- Bypass firewalls that only allow “established” connections (fancy way to say that they block incoming packets with the “SYN” flag)
  - Doesn't work if the firewall builds a table of outgoing connections
    - ❖ e.g., Network Address Translation, a.k.a. IP masquerading
- Response is a RST packet whether the port is closed or open
- Allows attackers to find out which IP addresses are in use, similar in function to an ICMP ping

## TCP FIN Scans

- RFC says:
  - open port, do not respond
  - closed port, respond with RST/FIN
- Some implementations respond with a RST on open ports
- Another way to map services on a host

## Defending Against Scans

- Issue: spoofed IP addresses
  - Option 1: Don't reply
    - ❖ "Stealth" does not increase bandwidth consumption
  - Option 2: "Active" defense
    - ❖ In a SYN scan, if you send a SYN/ACK for every packet, you could force the attacker to complete the connection to gain information
      - Makes it more difficult to spoof the source IP
      - Slows down scanners
        - » But if 1 in 100 packets is not spoofed, it slows down your server 100 times more than the scanner!
      - However:
        - » Increases traffic, bandwidth consumption
        - » May have undesired effects
        - » Replies sent to spoofed IP addresses
        - » Are you unwittingly attacking them?

## Mini-Lab

- You will use nmap to discover which ports are open on the lab machines
- Start a root shell (vulnerability assessment)
- Run "nmap -sO 127.0.0.1"
  - Result lists the protocols running on your machine
- Run "nmap -sS 127.0.0.1"
  - Does a SYN scan
- Try other options
  - -sF FIN scan
  - -sX Christmas scan (FIN, URG, PUSH)
  - -sN NULL (no flags)
- Try other machines (isolated networks are nice)

## Mini-Lab Results

- Did you find an open X11 port (6000)?
  - What if you telnet to it?
  - What if attackers open lots of connections to that port?
    - ❖ Troll attack
- Windows server
  - What information was available from port 139?
- Did you notice the ICMP scanning options (man nmap)?

## SYN Flood

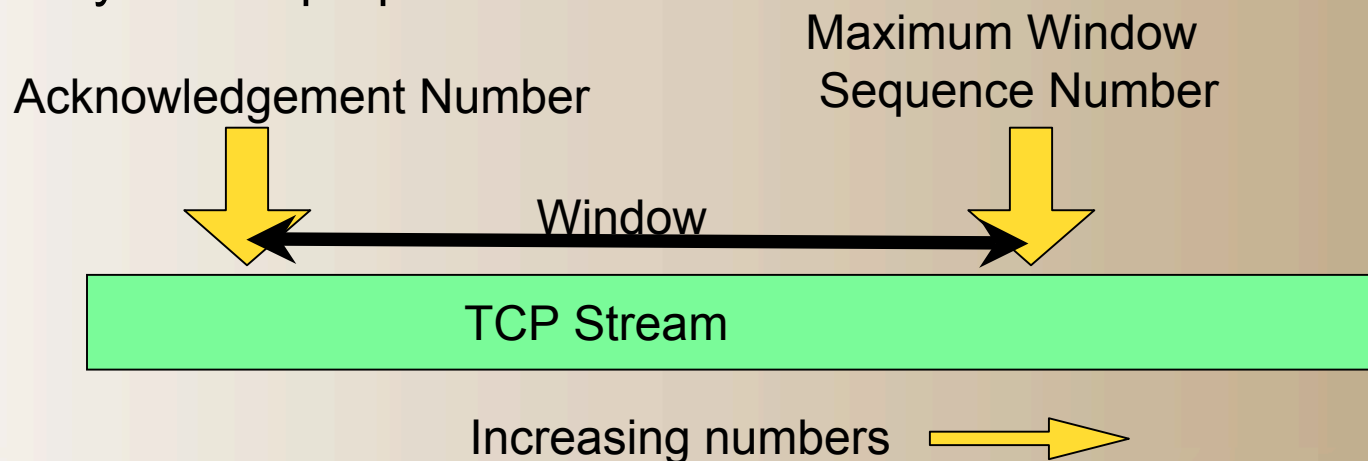
- Denial of service when an attacker sends many SYN packets to create multiple connections without ever sending an ACK to complete the connection, aka SYN flood.
  - CVE-1999-0116
  - Keeping track of each half-open connection takes up resources

# TCP Reliability

- A TCP connection is a *stream*
- Each TCP packet contains a *stream segment*
- A sequence number is associated to each byte
  - Packets have a single field for the sequence number
    - ❖ e.g., refers to the sequence number of a specific byte, according to a convention described in the RFC
- An ACK is required for each byte
  - If an ACK is not received in a certain amount of time, data is retransmitted
  - An ACK packet serves as an ACK for all bytes up to the byte indicated by the ACK's sequence number
- Receiver uses sequence numbers to correctly reorder segments and remove duplicates

# TCP Flow Control

- How much can a sender send at a time?
  - The more can be sent, the more efficient the network is
    - ❖ Fewer header bytes, media contention delays, etc...
- TCP "Window"
  - With every ACK, the receiver indicates how many more bytes it is prepared to receive



# TCP Sequence Numbers

- Every new connection gets a new initial sequence number (ISN)
  - For both sides of the connection
  - ISNs are exchanged (jargon: streams are "synchronized") in the initial SYN handshake
- TCP packets with sequence numbers outside the window are ignored
  - This makes attacks on TCP applications harder than if they used UDP

## Finding Sequence Numbers (For Attackers)

- Sniffing
- MAC address man-in-the-middle attack
- Source-routed IP packets (to setup a M.I.M. attack)
  - See the slides on routing
- ICMP redirects
- If the above is not possible, try to predict the initial sequence number
  - Connect yourself, examine how sequence numbers are generated (e.g., last one + 128 000)
  - Make a guess based on observations

# Randomness Visualization

- Strange attractors
  - Zalewski 2001, 2002 "Strange Attractors and TCP/IP Sequence Number Analysis"
- Given a sequence of numbers  $s[n]$  compute:
  - $x[n] = s[n-2] - s[n-3]$
  - $y[n] = s[n-1] - s[n-2]$
  - $z[n] = s[n] - s[n-1]$
- These are the x,y,z coordinates of a point
  - Plot them to see hidden dependencies

# Cisco IOS 12.2 (Zalewski 2002)

```
x0 y0 z1 vis 2147483648 (32) 0.00000 64389/100000 (64.3890%)
```



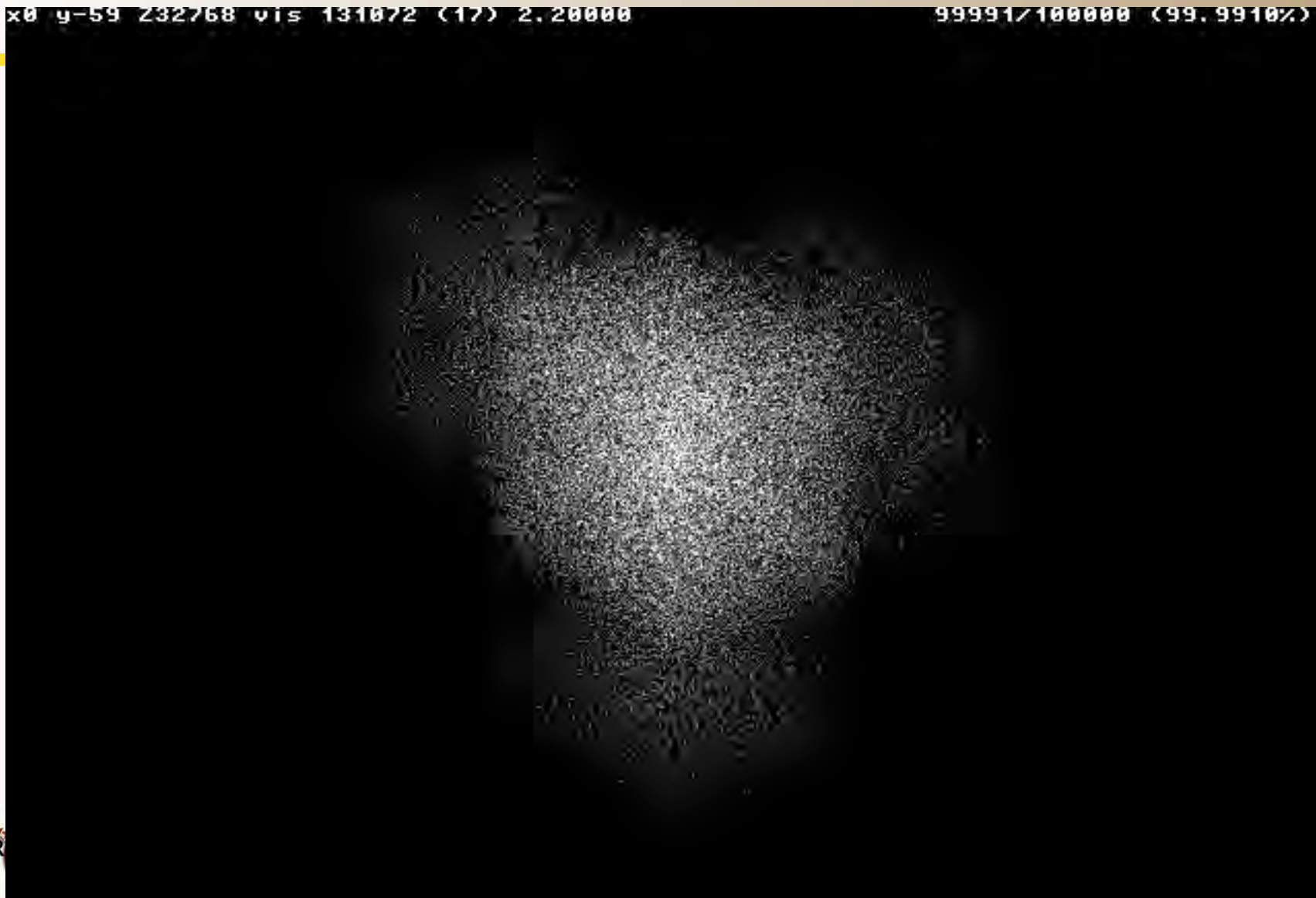
# IRIX (Zalewski 2002)

```
x0 90 216384 01s 262144 (18) 2.20000 99988/100000 (99.9880%)
```



# Windows XP (Zalewski 2002)

x0 9-59 232768 vis 131072 (17) 2.20000 99991/100000 (99.9910%)



# ISN Vulnerabilities

- Predictable
  - Symantec Raptor Firewall 6.5 and 6.5.3, Enterprise Firewall 6.5.2 and 7.0, VelociRaptor Models 500/700/1000 and 1100/1200/1300, and Gateway Security 5110/5200/5300 generate easily predictable initial sequence numbers (ISN), which allows remote attackers to spoof connections.
    - ❖ CAN-2002-1463
  - Cisco switches and routers running IOS 12.1 and earlier produce predictable TCP Initial Sequence Numbers (ISNs), which allows remote attackers to spoof or hijack TCP connections.
    - ❖ CVE-2001-0288
  - etc...

# TCP RST Flag

- TCP reset (RST) flag is used to abort TCP connections, usually to signify an irrecoverable error
  - Receiver deletes the connection, frees data structures
- RST messages are accepted only if they fit inside the sequence number window
  - Prevents delayed RST messages from previous connections to affect the current connection

# TCP RST Attack

- Send a RST (TCP RESET flag) packet with a spoofed IP address to either side of a valid connection
  - Need to guess a sequence number inside the appropriate window
    - ❖ Or sniff traffic to know which number to use
  - The range can be guessed fairly efficiently for RST attacks
  - Sequence numbers: 32 bits
  - Window size: up to 16 bits
  - Number of guesses  $32-16 = 16$  bit address space
    - ❖ 65535 RST attempts, ~ 4 min on DSL connection
    - ❖ Faster connection or zombies, faster RST
    - ❖ This is the brute force RST attack

# TCP Session Hijacking

- Idea: all that's required to mess up someone else's TCP session is guessing or knowing the sequence numbers for their connection.
  - Only need to fall within the needed range, exact guess not needed
- Send a spoofed IP packet, with a TCP payload that inserts data
- Blast the legitimate client off the net
  - Replies are still sent to client but client is incapacitated
  - You do not get to see replies: “blind” hijacking
    - ❖ Unless you can sniff traffic, in which case the sequence numbers to use are also known

## The TCP MD5 Signature Option

- Introduced as a way to protect BGP (see later) against RST attacks
- Hash algorithm
  - From some data, produce a number such that:
    - ❖ Any change in the original data would change the number
    - ❖ From the number, the data can't be reverse-engineered
  - MD5 is a popular hash algorithm
- Used for Integrity verification
  - e.g., Tripwire
- Problem: in a message, what prevents an attacker from changing the message **and** the hash?

## Solution

- Hash (data + secret)
  - The secret is unknown to the attacker, so the attacker can't produce a valid hash
- Append Hash (TCP packet + secret) as a TCP option
- Packets with invalid hashes are discarded
  - Makes the RST attack very difficult!

## Example Hash

- Hash without secret: (try it in a shell)
  - % md5  
This is my data <ctrl-D>  
8ceda0d1f9a39e19bc38d9a392754de2  
%
- Attacker can replace both the data and hash!
- Hash with secret:
  - % md5  
This is my data  
<some secret not included in packet><ctrl-D>  
c3a8cccc13ab218feca05cb5832c71a1  
%
  - What secret did I use?
    - ❖ You can't make a valid hash without knowing it

## Limitation

- You have to distribute the secret between N people
- Anyone can leak the shared secret
  - Who do you tell the secret to?
  - Really have to use a different secret with each other host your application talks to
    - ❖ Need  $N^2$  secrets
    - ❖ Very hard to manage
- Useful only for applications with a limited number of participants (e.g., core internet routers)

## Defenses

- One-time passwords ineffective because the session is hijacked after authentication...
- Only complete defense: encrypted traffic like ssh, IPSEC, IPv6
  - shared encryption key
- Helps to use strongly random ISNs
  - Even so, accepted range means limited number of guesses
- <http://www.cert.org/advisories/CA-2001-09.html>
- <http://www.sans.org/rr/threats/hijack.php>
- U.S. CERT Tech Alert TA04-111A.html

## Question

- Which is more difficult to attack?
- a) UDP
- b) TCP
- c) ICMP
- d) IP reassembly

## Question

- Applications that encrypt their data and validation checksum and use TCP, such as SSL (https) in browsers, are vulnerable to:
- a) RESET attacks
- b) Data injection attacks
- c) Session hijacking
- d) Eavesdropping attacks
- e) IP fragmentation attacks

## Question Answer

- Applications that encrypt their data and validation checksum and use TCP, such as SSL (https) in browsers, are vulnerable to:
  - **a) RESET attacks**
  - b) Data injection attacks
  - c) Session hijacking
    - Even if TCP session has injected data, the application should reject it due to the encrypted checksum
  - **d) Eavesdropping attacks**
  - ~e) IP fragmentation attacks**
    - depending on IP implementation, not really an application vulnerability

## Question

- UDP Ping-Pong is possible due to:
- a) IP fragmentation attacks
- b) Initial sequence number spoofing
- c) MAC address spoofing
- d) IP address spoofing

## Question

- UDP Ping-Pong is possible due to:
- a) IP fragmentation attacks
- b) Initial sequence number spoofing
- c) MAC address spoofing
- **d) IP address spoofing**

## Question

- To offer resistance to brute force RST attacks, you could:
  - a) Use application-level encryption
  - b) Use the largest TCP window possible
  - c) Use a small TCP window
  - d) Use unpredictable initial sequence numbers

## Question

- To offer resistance to brute force RST attacks, you could:
  - a) Use application-level encryption
  - b) Use the largest TCP window possible
  - **c) Use a small TCP window**
    - **The statistical attack tries to get a RST packet inside the window (but this results in less efficient communications)**
  - d) Use unpredictable initial sequence numbers
    - This helps against session hijacking, and predictive RST attacks. In the statistical RST attack, the attacker has given up on identifying your next ISN by prediction using prior ISNs

## Question

- A SYN flood attack works by:
  - a) Making computers drop established connections
  - b) Preventing new connections by consuming server resources
  - c) Injecting data in established connections
  - d) Creating a loop between services

## Question

- A SYN flood attack works by:
  - a) Making computers drop established connections
  - **b) Preventing new connections by consuming server resources**
  - c) Injecting data in established connections
  - d) Creating a loop between services

## Question

- UDP is attractive as an exploitation vector because
  - a) It's fast
  - b) It consumes resources to keep track of connections
  - c) It can change routing tables
  - d) It's anonymous through IP spoofing

## Question

- UDP is attractive as an exploitation vector because
  - a) It's fast
  - b) It consumes resources to keep track of connections
  - c) It can change routing tables
  - **d) It's anonymous through IP spoofing**

# Network Address Translation

- NAT, a.k.a. masquerading, allows sharing an internet connection through a single IP address
- IP addresses and checksums are replaced on-the-fly
- NAT maintains a table of mappings:
  - Internal IP, port  $\Leftrightarrow$  External IP, port
  - Incoming packets are translated back if there's an existing mapping, dropped otherwise
  - Mappings are added only for outgoing packets

## Question

- What will happen if someone tries to do a SYN TCP scan from inside a NAT firewall?
- a) Victims won't know who did the scan and can't retaliate
- b) Victims will be under a SYN flood attack
- c) The NAT mapping table will overflow, possibly preventing other users from accessing the web

## Question

- What will happen if someone tries to do a SYN TCP scan from inside a NAT firewall?
- a) Victims won't know who did the scan and can't retaliate
- b) Victims will be under a SYN flood attack
- **c) The NAT mapping table will overflow, possibly preventing other users from accessing the web**

# TCP Reflection Attacks

- TCP is reliable, so it resends packets until an ACK is received
  - Amplification
- Scenario
  - Send a SYN packet to a server, using victim's IP address (IP spoofing)
  - Server "replies" with a SYN/ACK several times
    - ❖ Server may be on a "big pipe" and able to flood victim
      - e.g., BGP routers on internet backbone
  - Reflection makes it more difficult to trace
  - SYN packets sent by zombie DDoS agents
  - If SYN/ACK packets are filtered, victim can't connect to outside server of same port!

## Services Targetted

- ssh
- DNS
- etc...
- Additional technique: Sequence number prediction
  - Request large file transfers to victim
    - ❖ Another form of amplification
- Reference: Paxson 2001 "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks"
  - <http://www.icir.org/vern/papers/reflectors.CCR.01/reflectors.html>

## Variant Reflection Attack

- IP has a new option, traceroute (RFC 1393)
- Each router on the route is supposed to send an ICMP Traceroute (type 30) message back to the "source"
  - Amplification
- Spoof the IP of the source, victim is overwhelmed (DDoS)
- Can be done with any IP packet (UDP, TCP, ICMP, IGMP...)

## Question

- In a TCP reflection attack, how do you protect a server against DoS?

## Answer

- Filter incoming SYN/ACKs, since the server isn't trying to make outgoing connections, but it may impair the capability to remotely manage the computer, or to download software updates
  - ISP should do the filtering to preserve the bandwidth of the link to the ISP

# Questions or Comments?

§

## About These Slides

- You are free to copy, distribute, display, and perform the work; and to make derivative works, under the following conditions.
  - You must give the original author and other contributors credit
  - The work will be used for personal or non-commercial educational uses only, and not for commercial activities and purposes
  - For any reuse or distribution, you must make clear to others the terms of use for this work
  - Derivative works must retain and be subject to the same conditions, and contain a note identifying the new contributor(s) and date of modification
  - For other uses please contact the Purdue Office of Technology Commercialization.
- Developed thanks to the support of Symantec Corporation



# **Pascal Meunier** **pmeunier@purdue.edu**

Contributors:

Jared Robinson, Alan Krassowski, Craig Ozancin, Tim Brown, Wes Higaki, Melissa Dark, Chris Clifton, Gustavo Rodriguez-Rivera

Thanks to Michal Zalewski for the permission to use images

