Announcements

• Lab 5 is out
  – Due Thursday, December 3rd

• Layered on top of Lab 3 (sr)
  – Pass a command flag (-nat) to turn on NAT behavior

• Lab 3 grade = max(lab 3 grade, lab 5 grade)
Overview

- Basic NAT functionality
- ICMP Requirements
- TCP Requirements
- General NAT processing logic
- Suggestions
NAT

• Network Address Translation

• Translates private IP addresses to facilitate Internet communication
  – 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

• Single device with single IP address
  – Hides details of internal network
  – But interferes with many applications
“Reverse” NAT

• myth (you) is behind NAT

• Distinguish internal (eth0) and external (eth1) by interface name

• Translate packets from myth (VNS firewall) so that it appears the NAT sent them
ICMP Requirements

• Support echo requests/replies

• Echo requests are external host independent
  – Using the same query identifier to two different hosts will preserve mapping
  – If A sends an ICMP request with id q1\rightarrow q1' to B and another request with id q1\rightarrow q2' to C, then q1'==q2'.

• Do not timeout ICMP query mappings for at least 60 seconds
TCP Requirements

1. Endpoint-Independent Mapping behavior for TCP
   - Same translation \((X1:x1) \rightarrow (X1’:x1’)\) for packets destined to any external host
   - UNSAF: Unilateral Self-Address Fixing mechanism

2. Support all valid sequences of TCP packets
   - TCP implementations should work

3. Endpoint-Independent Filtering behavior for TCP
   - Like Endpoint-Independent Mapping, just for accepting inbound packets from external hosts
TCP Requirements

4. Don’t respond to inbound SYN for at least 6 seconds. Drop if outbound SYN received, send Port Unreachable otherwise
   - Used for supporting simultaneous open
   - Compromise to have this support and signal error for invalid SYN

5. Abandon idle TCP connections after 2 hours 4 minutes
   - Rationale: Default keep-alive of 2 hours and transitory period (open/close) of 4 minutes
   - Can drop or send RST packets for non-SYN pkts with no mapping
TCP Requirements

6. No port assignment behavior of port overloading for TCP
   - Disallow different internal endpoints from using the same mapping
   - This means for \((X1:x1) \rightarrow (X1’:x1’)\) and \((X2:x2) \rightarrow (X2’:x2’)\), \((X1’:x1’) \neq (X2’:x2’)\)

7. Support hairpinning for TCP of type “External source IP address and port”
   - Rewrite source IP and port when receiving packet from internal host with a mapping
Hairpinning

Mapping
Y:y → Y':y'
X:x → X':x'

csr IP, port - X:x
dst IP, port - Y':y'

eth1: 171.67.236.20

eth0: 171.67.236.16

src IP, port - X':x'
dst IP, port - Y:y

myth

X:x

myth

Y:y
General Logic

• Check whether packet is inbound or outbound
• Determine if it is ICMP or TCP
• If outbound, add a globally unique mapping
• If inbound, check for existing mapping.
  – If none, discard (unless TCP SYN or hairpinning)
General Logic

• Rewrite IP src/dst
  – Don’t forget to recompute checksum
• Rewrite ICMP identifier/TCP port
  – Recompute checksum again
  – TCP checksum covers *pseudoheader* and payload
• Reuse router logic to determine how to forward packet
• Don’t worry about UDP
Threads

• Spawn a thread to handle timing out NAT entries
  – Similar to ARP cache

• Synchronize access to shared data
  – NAT mappings
  – Locks

• Create thread in sr_router.c
  – Takes a pointer to a C routine. This is where you implement timeout logic.

• Can rely on main program exit to terminate thread
Data Structures

• Need to store NAT mappings
  – Linked list is fine, O(n) traversal
  – Keep a time field to remember when a mapping was last used

• Need to remember used ICMP identifiers and used port numbers
  – Separate structures for identifier and port number
Implementation Suggestions

• Implement NAT code in separate files (e.g. sr_nat.h, sr_nat.c)
  – Don’t forget to update the Makefile

• Handle command line flags in sr_main.c

• Create necessary NAT data structures in sr_instance (sr_router.h)
  – Initialize in sr_router.c
Other Suggestions

• Work on ICMP first and then TCP
  – Note that ARP is unchanged

• Save logfile (-l logfile to ./sr) and examine packet flow in Wireshark/tcpdump

• Start early – report VNS issues to staff list and VNS admin (dgu@cs.stanford.edu)
Upcoming Updates

• Reference binary for comparison
  – Will be released next week, accessible from /usr/class/cs144/bin

• New topology for testing
  – Most likely will be nested NATs

• Web server will likely be updated to show observed IP address/port on home page
Questions?

$32 + 16 > 128$