CS514: Intermediate Course in Computer Systems

Lecture 20: Nov 5, 2003
“Firewalls and NATs”

Summary of Lecture

Firewalls and NATs
- Typical firewall and NAT deployment
- Firewall capabilities and configuration
- Problems for applications developer
- Getting through firewalls
  - Windows Media case study
- How NATs work
- Getting through NATs
  - Continue next lecture…
Site with no firewall

ISP Router

Link (T1 etc.)

Site Router

Site Network

Site with firewall

ISP Router

Firewall

Site Router

Site Network
Site with firewall

Well, not really!

DMZ (“De-Militarized Zone”)

DMZ: Network outside of Site security perimeter used to deploy firewall(s) and publicly available services (Web, FTP, DNS, etc.)
Various DMZ deployments are possible

ISP Router
  Site Router
    Firewall/NAT
      Site Router

History: Firewalls were rogue components

- Firewall/DMZ architecture never part of the “official” Internet Architecture
  - Purely a commercial creation
  - Distrusted by IAB (Internet Architecture Board)
- “Crunchy on the outside, soft on the inside”
  - “All security should be end-to-end”, etc…
Firewalls not just protection from outside attackers

- Bandwidth control
  - Block high bandwidth applications
  - Pointcast, Napster
- Employee network usage control
  - Block games, pornography, non-business uses
- Privacy
  - Don’t let outside see what you have, how big you are, etc.
  - Similar to making corporate phone directory proprietary

Firewall functions

- Dropping packets
  - According to 5-tuple and direction of packet (incoming or outgoing)
    - Recall: 5-tuple = src/dst address, src/dst port, protocol
    - According to “conversation”
      - Multiple related flows, like FTP, SIP
    - According to higher-layer info (i.e. URL)
- Steering packets/messages
  - To other filters, like spam filter, virus checker, HTTP filter, etc.
- Logging flows and statistics
Simple firewall policy configuration

<table>
<thead>
<tr>
<th>Source</th>
<th>Dest</th>
<th>App</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>any-inside</td>
<td>dmz-mail</td>
<td>SMTP</td>
<td>allow</td>
</tr>
<tr>
<td>any-inside</td>
<td>any-outside</td>
<td>SMTP</td>
<td>drop</td>
</tr>
<tr>
<td>any-inside</td>
<td>any-outside</td>
<td>HTTP</td>
<td>allow</td>
</tr>
<tr>
<td>any-inside</td>
<td>any-outside</td>
<td>FTP</td>
<td>allow</td>
</tr>
<tr>
<td>any-inside</td>
<td>any-outside</td>
<td>any</td>
<td>drop</td>
</tr>
<tr>
<td>any-outside</td>
<td>any-inside</td>
<td>any</td>
<td>drop</td>
</tr>
</tbody>
</table>

Conversations

- FTP consists of two flows, control flow and data flow
- Firewall must be smart enough to read control flow, identify subsequent data flow
- True for SIP as well
Stateful and stateless firewalls

- Original firewalls were stateless
  - Maintain static filter list, but no per flow state
  - For TCP, only look at SYN
    - Means that non-SYN TCP packets are allowed even if should be blocked
  - No concept of conversation
- Modern firewalls are typically stateful
  - Maintains dynamic list of all allowed flows
  - Better capability, harder to scale

Routing-based or callout-based steering (1/2)

- Callout-based:
  - User-customized functions may be called at specific checkpoints
    - i.e. after each individual email in an email stream
    - after each HTTP GET
  - These callouts can operate on the firewall box, or send messages to another box
    - i.e. after each mail message, local callout looks for attachments, and if found sends mail to a virus checker
Routing-based or callout-based steering (2/2)

- **Routing-based**
  - Packets matching policy rule sent to another box
  - Destination address may be modified to that of the box
    - if box is not promiscuous

Problem for app developer

- Obviously, your application may be blocked by the firewall
- Two basic strategies:
  1. Hide the application inside HTTP
  2. Make it easy for the firewall administrator to allow your application
- Which strategy you use depends on why the app is being blocked
Intentional versus unintentional blocking

- Unintentional blocking:
  - Blocking is a side effect of a broader policy
  - i.e., all UDP blocked, even though in principle the admin has no problem with your application

- Intentional blocking:
  - The admin knows of your application, and really does want to block it
  - i.e. Napster

Strategy for intentional blocking

- Long term, this is a hard battle to win
  - Can try to hide everything in what looks like normal HTTP, but the administrator can fight this in various ways:
    - Block on specific URLs
    - Block on specific IP addresses
    - Disallow the application on the client computers

- Better to solve the network admin’s concerns
  - Allow a caching proxy in the DMZ
  - Although this didn’t work for Pointcast…. 
Strategy for unintentional blocking

- “Hide” the application in HTTP
- But also allow the application to run “natively” if you get performance benefits
  - Make firewall configuration for allowing the application as simple as possible
  - i.e. one or a small number of specific ports
  - Get the port blessed by IANA
    - Internet Assigned Numbers Authority

Case study: Windows Media

- Can run in four modes (from most to least efficient):
  1. IP multicast
  2. UDP
  3. TCP
  4. HTTP
- Windows media client will attempt to connect in the above order
- TCP firewall “holes” are simple to configure
  - TCP port 1755
  - Admin can specify which UDP ports
- Also allows a proxy in the DMZ
Windows Media client network configuration

Ethereal trace: First MMS stream
Ethereal trace: Second MMS stream

Speaking of weird protocol tunneling….

- My favorite is IP over DNS
- This is actually a “legitimate” example
  - (Wait until end of class for why)
Network Address Translation (NAT)

- NAT invented to solve the address depletion problem
  - In early 1990’s, we thought we’d run out of IPv4 addresses by mid-to-late 1990’s
  - Currently about ½ of IPv4 addresses are allocated (out of total 4 billion)
- No longer an address depletion “crisis”
- Two reasons for this:
  - Tougher allocation policies
  - NAT

Original NAT design: Global address shared over time

List of global IP addresses
20.1.1.1
20.1.1.2
....
20.1.1.10
Original NAT design: Global address shared over time

List of global IP addresses
20.1.1.1
20.1.1.2 ⇔ 10.1.1.1
....
20.1.1.10 ⇔ 10.1.1.254

Private Network
10.1.1.1/30.1.1.1
10.1.1.2/30.1.1.1
....
10.1.1.254/30.1.1.1

NAT
10.1.1.1/30.1.1.1
10.1.1.2/30.1.1.2
20.1.1.10/30.1.1.2

Global Internet
30.1.1.1
30.1.1.2
Original NAT design: Global address shared over time

- Original NAT predates the web
- Assumption was that one global address could support tens of hosts
  - Occasional FTP, etc.
- Web changed the usage model
  - More frequent global accesses
  - NAT was enhanced to allow addresses to be shared at the same time
  - Port translation (sometimes called NAPT)

Current NAT design: Global address shared at one time

- One global IP address
  - 20.1.1.1
- Private Network
- List of port assignments
- Global Internet
  - 30.1.1.1
  - 30.1.1.2
- NAT
  - 10.1.1.1
  - 10.1.1.2
  - 10.1.1.254
Current NAT design: Global address shared at one time

List of port assignments
10.1.1.1 ⇔ 1111(2345)
10.1.1.254 ⇔ 1112(2345)

One global IP address
20.1.1.1

List of port assignments
10.1.1.1/30.1.1.1 2345/6789
10.1.1.2/30.1.1.2 2345/7890

Current NAT design: Global address shared at one time

List of port assignments
10.1.1.1/30.1.1.1 1111/6789
10.1.1.2/30.1.1.2 1112/7890

One global IP address
20.1.1.1/30.1.1.1 6790/4444

List of port assignments
10.1.1.1/30.1.1.1 1113/4444
10.1.1.1/30.1.1.2 1112/7890

Problems with NAT

- Hard to make incoming connections
  - But will show you how in next lecture
  - This marketed as a feature of NAT!
- Some applications break
  - Those that carry IP address in upper layers
  - Less of a problem than it used to be
    - NAT boxes translate IP addresses in upper layers for common applications
    - Application designers now know not to put IP addresses in the upper layers

(Unexpected) advantages of NAT

- Isolates site from global addressing
  - Can change ISPs without renumbering
- Privacy
  - ISPs could otherwise charge you per host
  - Hard to tie IP address to user
  - Outside can’t deduce how many hosts you have
- Fun to irritate IETF end-to-end purists
- NAT is marketed as security enhancement, but this is mainly a bogus claim
Attempts to fix NAT

- RSIP (Realm Specific IP)
  - IETF work
  - Host can request an address assignment from the NAT box
  - Didn’t go anywhere
- Microsoft UPnP (Universal Plug and Play)
  - Similar to RSIP (I think)
- STUN (Simple Traversal of UDP through NAT)
  - Bad name…try searching for it with Google!
  - Simple method for host to learn what port it got assigned (transparent to NAT box)
  - Then application can use this knowledge as it sees fit

I like STUN

- I think it will succeed
- I think it will be another nail in the coffin of IPv6
- I wish I had thought of it
- More on STUN next class
IP over DNS

- Wireless LAN service in Finland
- Used HTTP “captive portal” to charge users
  - First HTTP access would be steered by firewall to a billing application
  - This allows billing without new software in client host
  - Once user pays, firewall allows all packets
- But, before client can do HTTP, it needs to get a DNS reply first
  - So firewall always allowed DNS to go through
- By tunneling IP over DNS, users could get free WLAN access!