CS419: Computer Networks

Lecture 8: Mar 28, 2005
Firewalls and NATs
I’m going to limit “network security” to three topic areas:

- Network access issues (user or host authentication, and VPNs)
- Site protection issues (firewalls and VPNs)
- Flow encryption issues (including key distribution)
  - IPsec at network layer
  - TLS or SSL or SSH at transport layer

I’m excluding application-level security, like S/MIME or secure email, as well as Kerberos
Site with no firewall

ISP Router

Link (T1 etc.)

Site Router

Site Network

CS419
Site with firewall

- ISP Router
- Firewall
- Site Router
- Site Network
Site with firewall

Well, not really!

(Nothing is this simple!)
DMZ ("De-Militarized Zone")

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

ISP Router

Firewall/NAT
Various DMZ deployments are possible
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…
Firewall model held up well until recently

- Email viruses and laptops now cause havoc
  - Firewalls scan incoming email, but laptops bypass firewalls
- Nowadays sites are proactive about what can attach to the internal network
  - Newly attached hosts are scanned for latest virus software and profiles
  - More and more, internal switches have firewall functionality, monitor all traffic!
Firewalls not just protection from outside attackers

- Bandwidth control
  - Block (or choke) high volume, non-critical applications
  - Kazaa

- 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, email attachments)

- **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
Firewall arms race

- Firewalls make it hard to introduce new applications
  - Because firewall rules tend to err on the side on prevention
- As a result, many new apps are built over HTTP
  - Or at least can fall back on HTTP if better performing protocols are blocked
  - Firewalls respond by looking deeper into HTTP/HTML, but this is hard
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

```
$ ã'è MMS ← « ³³³³ » NSPlayer/7.1.0.30
55; {D4C55213-364F-4CF6-A7F6-90F4DFBA98F8}
; Host: wm.sony.global.speedera.net $ ã'p
MMS Ū $ « ³³³³ » ëëëë ê? $ ³ 4.1.0.39
23 $ ã'° MMS $ Ï Hâzï®GÑ? $ ūëë $ ã'°@ MMS Ū $ $ ūëë $ Éö MMS ê $ ôyÔxé& ã @ $ ūëëyy鈥ї
MMS Ū $ $ ūëë $ Eö MMS ê $ ôyÔxé& ã @ $ ūëëyy鈥ї
$ Ū Funnel Of The $ ã'° MMS
| | | | \128.84.99.231\UDP\2366 3 | ã'°@ MMS
| | | | | Zd;βO•@® $ ū¥¥¥ wm.sony.global/PearlJ
| | | | | am/saveyoufullvid_100.wmv $ ã'°
| | | | | MMS $ | .........
```
Ethereal trace: Second MMS stream

```
\128.84.99.231\TCP\2367
```

Funnel Of The
Speaking of weird protocol tunneling…. 

- My favorite is IP over DNS
- This is actually a “legitimate” example
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!
NATs and firewalls

- NAT and firewall functions typically co-exist in the same box.
- NAT is marketed as enhancing security.
  - There may be a smidgen of truth to this, but in fact it doesn’t enhance security much beyond what a firewall can do.
  - Probably reduces problems with configuration errors.
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
10.1.1.2
....
10.1.1.254

10.1.1.1/30.1.1.1
10.1.1.2/30.1.1.2

NAT

Global Internet

20.1.1.2/30.1.1.1
20.1.1.10/30.1.1.2

30.1.1.1
30.1.1.2
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 ⇔ 10.1.1.2

NAT

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

Global Internet
20.1.1.2/30.1.1.1
20.1.1.10/30.1.1.1
30.1.1.1
30.1.1.2

10.1.1.1
10.1.1.2
...
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

10.1.1.1
10.1.1.2
...?
10.1.1.254

NAT

30.1.1.1
30.1.1.2
Current NAT design: Global address shared at one time

One global IP address
20.1.1.1

List of port assignments
10.1.1.1 ⇔ 1111(2345)
10.1.1.254 ⇔ 1112(2345)
Current NAT design: Global address shared at one time

- Global Internet
- Private Network

List of port assignments:
- 10.1.1.1 ⇔ 1111(2345)
- 10.1.1.254 ⇔ 1112(2345)
- 10.1.1.1 ⇔ 1113(6790)

Port assignments:
- 10.1.1.1/30.1.1.1
- 20.1.1.1/30.1.1.1
- 20.1.1.1/30.1.1.1
- 20.1.1.1/30.1.1.1
- 20.1.1.1/30.1.1.1
- 20.1.1.1/30.1.1.1

Networks:
- 10.1.1.1
- 10.1.1.2
- 10.1.1.254

Global address:
- 20.1.1.1

Private Network:
- 10.1.1.1/30.1.1.1
- 20.1.1.1/30.1.1.1
- 20.1.1.1/30.1.1.1

Global Internet:
- 20.1.1.1/30.1.1.1
- 30.1.1.1
- 30.1.1.2
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 :(
Attempts to fix NAT (1/2)

- RSIP (Realm Specific IP)
  - IETF work
  - Host can request an address and address+port assignment from the NAT box
  - Didn’t go anywhere

- Microsoft UPnP (Universal Plug and Play)
  - Broad initiative to allow cross-vendor plug-and-play in local network environment
    - Auto-configure into net, advertise its capabilities
  - NAT aspect: Client can learn of address/port mappings from NAT box, add new port mappings
  - I don’t know if this is taking off or not
Attempts to fix NAT (2/2)

- midcom (middlebox communications)
  - IETF working group
  - Broad effort to deal with all kinds of (now opaque) middle boxes (NATs, firewalls, Intrusion Detection Systems (IDS), etc.)
  - Usual standards committee trashing about

- 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

- RFC 3489
- I think it will succeed
  - Note that, of these options, STUN is the only one that doesn’t require NAT box cooperation
  - This is a big win…
- I think it will be another nail in the coffin of IPv6
- I wish I had thought of it
## Types of NAT behaviors

<table>
<thead>
<tr>
<th>Type</th>
<th>Port assignment policy</th>
<th>Firewall policy for incoming packets (from dest address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full cone</td>
<td>Same global addr and port for every internal address and port (from a given internal host)</td>
<td>Accept all flows to assigned address and port from any dest address</td>
</tr>
<tr>
<td>Restricted cone</td>
<td></td>
<td>Accept if internal packet previously sent to dest address</td>
</tr>
<tr>
<td>Port-restricted cone</td>
<td></td>
<td>Accept if internal packet previously sent to dest address and port</td>
</tr>
<tr>
<td>Symmetric</td>
<td>Different addr/port for every flow</td>
<td></td>
</tr>
</tbody>
</table>
What STUN does

- Tells you if you are behind a NAT
  - If so:
    - Tells you the assigned address(es) and port(s)
    - Tells you what type of NAT
  - If not:
    - Can still tell you what kind of firewall you are behind
      - (UDP blocking, symmetric UDP)
Packet can’t come in until NAT box has mapping
STUN server sees the global addr/port, and informs host

What is my UDP addr/port?

Your UDP addr/port is 1.1.1.1:1234
Steve and Bob register with globally addressed server

App: I’m steve
UDP: 1234
IP: 1.1.1.1

App: I’m bob
UDP: 5678
IP: 2.1.1.1
Server tells Steve and Bob each other’s NAT mapping

I want to talk to Bob

Steve is at 1.1.1.1:1234
Bob is at 2.1.1.1:5678
Steve sends “bubble packet” to create his mapping

Creates “pinhole” in Steve’s NAT

No “pinhole” here yet
Bob does the same, but this packet gets through

Creates “pinhole” in bob’s NAT
Steve and Bob can talk!
Limitations of this approach

- Doesn’t work with some kinds of NATs
  - NAT must always assign same external port to a given internal port
- Doesn’t work for TCP
  - Because TCP is *usually* asymmetric… expects a listener and a connecter
    - Windows OSs and some firewalls enforce this
  - We have a project to fix this problem
- Many corner cases (for instance, two hosts behind same NAT)
SIP with STUN (simplified)

Steve and Bob run STUN and register with respective SIP servers.

10.1.1.1 10.1.1.2
1.1.1.1 2.1.1.1

SIP Server
STUN Server
STUN Server
SIP Server

10.1.1.1
steve@sip.foo.com

2.1.1.1
bob@sip.bar.com

10.1.1.2
SIP INVITE message
To: bob@sip.bar.com
From: steve@sip.foo.com
SDP: 1.1.1.1:1234
SIP with STUN (simplified)

SIP 200 OK message
- To: steve@sip.foo.com
- From: bob@sip.bar.com
- SDP: 2.1.1.1:5678
SIP with STUN (simplified)

UDP packets flow directly

SIP Server

1.1.1.1

2.1.1.1

10.1.1.1

10.1.1.2

steve@sip.foo.com

bob@sip.bar.com
How to determine if NAT is restricted

- STUN server can send packets from two addresses and two ports
  - Primary and secondary
  - pA and pP, sA and sP
- STUN client can ask the STUN server to use the secondary port or address and port.
Keeping NAT assignments alive

- NAT box will time-out port assignment after inactivity (if UDP)
  - At end of TCP connection if TCP
- App must periodically send packets to keep NAT state alive
  - Every minute or so?
- Note that client can try to learn NAT box time-out value
  - But this takes time, and is prone to failure
What about this????

Diagram:

- STUN Client
- Private net
- NAT1
- Private net
- NAT2
- Global Internet
- STUN Server
- App Server
Don’t really want this…
And some NATs don’t allow it!
May use heuristics to decide if on same private network

- Peers have same global IP address
  - But this may not happen
- Peers have same domain name
  - Doesn’t mean peers are in the same private network though
- Doesn’t hurt (much) to try local address and global address
What about this????

This is the only choice. No way to learn these addresses.
Discovering STUN servers

Two ways:

- By address
- By name
  - By SRV record (preferred)
  - By A record (if SRV doesn’t work)
Stuff I didn’t talk about

- Before the query/reply, there is a security phase over TCP using TLS
  - The STUN server securely gives you a temporary name and password
- Other details to overcome security problems
Typical STUN deployment

- STUN Client
- NAT1
- NAT2
- Global Internet
- STUN Server

Your home NAT

ISP’s NAT
Basic operation: query/reply

STUN Client  private net  NAT1  private net  NAT2  Global Internet  STUN Server

What is my global address and port?

sA=10.1.1.1, sP=5555  sA=10.1.1.2, sP=6666  sA=20.1.1.1, sP=7777

dA=10.1.1.1, dP=5555  dA=10.1.1.2, dP=6666  dA=20.1.1.1, dP=7777

Your global address is 20.1.1.1, port is 7777
Use learned address/port to tell peer how to reach you

Open port 5555

(Note, must use same source port for app that was used with STUN to get same assignment from NAT box.)
Use learned address/port to tell peer how to reach you

I’m at 20.1.1.1:7777
Voila, it works!

STUN Client

Global Internet

STUN Server

App Server

NAT1

NAT2

dA=20.1.1.1, dP=7777

dA=10.1.1.1, dP=5555

dA=10.1.1.2, dP=6666
Unless NAT is restricted!

I don’t know about you...

dA=20.1.1.1, dP=7777

sA=30.1.1.1, sP=8888
If restricted NAT, must “punch hole” first

dA=30.1.1.1, dP=8888

Now I know about 30.1.1.1:8888
How to determine if NAT is restricted

What is my addr/port?

Your addr/port is A/P. Here is sA and sP.

This time reply from sA and sP.

Ok, here is my reply.

Blocked if restricted NAT.