Lecture 23: Firewalls

https://cs161.org

Announcements

- Discussion sections are online; we have a somewhat reduced schedule. We'll also record some of them. See Piazza.
- Office hours are online-only. See Piazza.
- Project 2 deadline extended until Tuesday, March 31, 11:59pm
- I welcome your questions and feedback during lecture in Zoom's text chat. (Mark your comments visible to Everyone if you're comfortable with that.)

Denial of Service

Algorithmic complexity attacks

- Attacker can try to trigger worst-case complexity of algorithms / data structures
- Example: You have a hash table.
  Expected time: O(1). Worst-case: O(n).
- Attacker picks inputs that cause table collisions.
  Time per lookup: O(n).
  Total time to do n operations: O(n^2).
- Solution? Use algorithms with good worst-case running time.
  - E.g., universal hash function guarantees that $Pr[h_k(x)=h_k(y)] = 1/2^b$, so hash collisions will be rare.

Application-Layer DoS

- Rather than exhausting network or memory resources, attacker can overwhelm a service's processing capacity
- There are many ways to do so, often at little expense to attacker compared to target (asymmetry)
- Defenses against such attacks?
- Approach #1: Only let legit users issue expensive requests
  - Relies on being able to identify/authenticate them
  - Note: that this itself might be expensive!
- Approach #2: Force legit users to "burn" cash
- Approach #3: massive over-provisioning ($$$)

DoS Defense in General Terms

- Defending against program flaws requires:
  - Careful design and coding/testing/review
  - Consideration of behavior of defense mechanisms
    - E.g. buffer overflow detector that when triggered halts execution to prevent code injection ⇒ denial-of-service

- Defending resources from exhaustion can be **really** hard. Requires:
  - *Isolation and scheduling mechanisms*
    - Keep adversary's consumption from affecting others
  - *Reliable identification* of different users

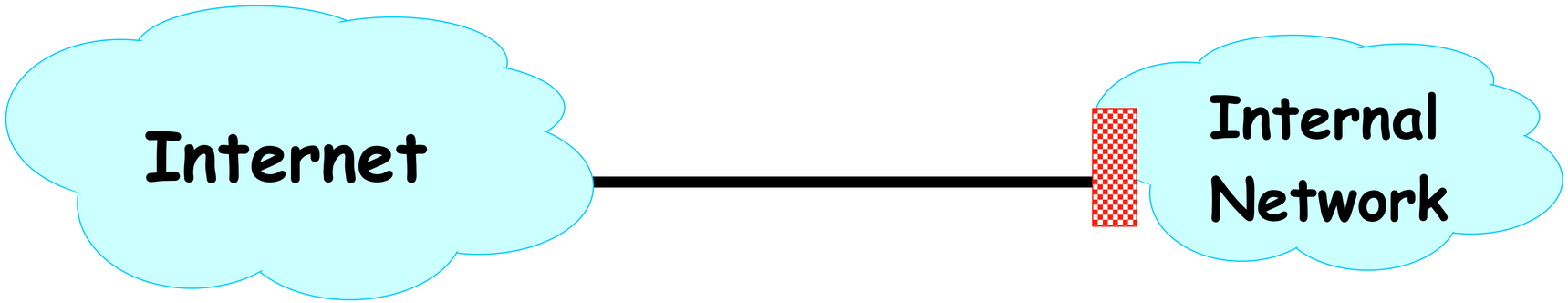
Firewalls

Controlling Networks … On The Cheap

- Motivation: How do you harden a set of systems against external attack?
  - Key Observation:
    - The more network services your machines run, the greater the risk
  - Due to larger attack surface
- One approach: on each system, turn off unnecessary network services
  - But you have to know all the services that are running
  - And sometimes some trusted remote users still require access
- Plus key question of scaling
  - What happens when you have to secure 100s/1000s of systems?
  - Which may have different OSs, hardware & users …
  - Which may in fact not all even be identified …

Taming Management Complexity

- Possibly more scalable defense: Reduce risk by blocking in the network outsiders from having unwanted access your network services
  - Interpose a firewall the traffic to/from the outside must traverse
  - Chokepoint can cover thousands of hosts
    - Where in everyday experience do we see such chokepoints?



Selecting a Security Policy

- Firewall enforces an (access control) policy:
  - Who is allowed to talk to whom, accessing what service?
- Distinguish between inbound & outbound connections
  - Inbound: attempts by external users to connect to services on internal machines
  - Outbound: internal users to external services
  - Why? Because fits with a common threat model. There are thousands of internal users (and we've vetted them). There are billions of outsiders.
- Conceptually simple access control policy:
  - Permit inside users to connect to any service
  - External users restricted:
    - Permit connections to services meant to be externally visible
    - Deny connections to services not meant for external access

How To Treat Traffic Not Mentioned in Policy?

- Default Allow: start off permitting external access to services
  - Shut them off as problems recognized
- Default Deny: start off permitting just a few known, well-secured services
  - Add more when users complain (and mgt. approves) ✓
- Pros & Cons?
  - Flexibility vs. conservative design
  - Flaws in Default Deny get noticed more quickly / less painfully

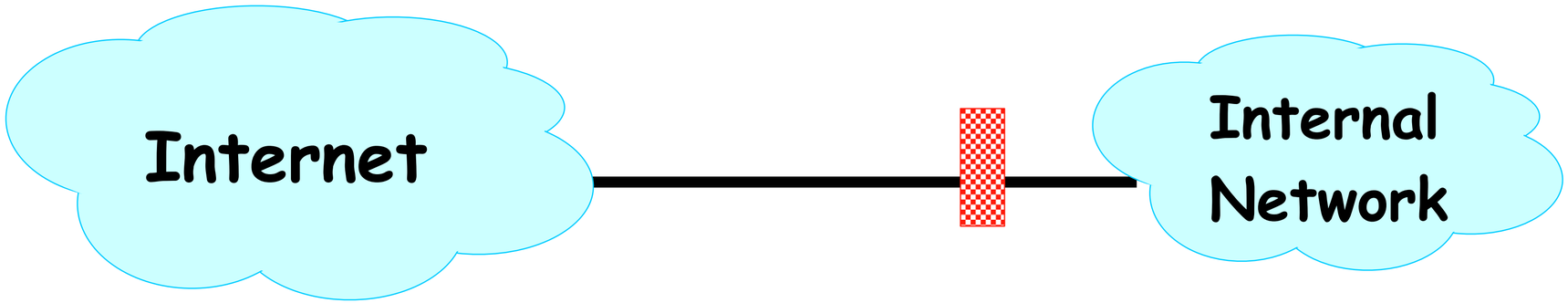
In general, use Default Deny

Stateless Packet Filter

- A stateless packet filter inspects each packet for certain filtering rules to determine whether to pass or block it (with no history)
- Simple policy: deny all inbound connections
  - Allow all outbound packets
  - Allow all inbound packets that are a reply... Do you see the problem?
- We can fake it for TCP connections, with a hack
  - Allow all outbound TCP packets
  - Allow all inbound TCP packets with ACK flag set
- We can't handle UDP connections

Stateful Packet Filter

- Stateful packet filter is a router that checks each packet against security rules and decides to forward or drop it
  - Firewall keeps track of all connections (inbound/outbound)
  - Each rule specifies which connections are allowed/denied (access control policy)
  - A packet is forwarded if it is part of an allowed connection



Example Rule

- `allow tcp connection 4.5.5.4:* -> 3.1.1.2:80`
  - Firewall should permit TCP connection that's:
    - Initiated by host with Internet address 4.5.5.4 and
    - Connecting to port 80 of host with IP address 3.1.1.2
  - Firewall should permit any packet associated with this connection
- Thus, firewall keeps a table of (allowed) active connections. When firewall sees a packet, it checks whether it is part of one of those active connections. If yes, forward it; if no, check to see if rule should create a new allowed connection

Example Rule

- `allow tcp connection *:*/int -> 3.1.1.2:80/ext`
  - Firewall should permit TCP connection that's:
    - Initiated by host with any internal host and
    - Connecting to port 80 of host with IP address 3.1.1.2 on external Internet
  - Firewall should permit any packet associated with this connection
  - The /int indicates the network interface.
- This is "Allow all outgoing web requests"

Example Ruleset

- `allow tcp connection *:*/int -> *:*/ext`
- `allow tcp connection *:*/ext -> 1.2.2.3:80/int`

- Firewall should permit outbound TCP connections (i.e., those that are initiated by internal hosts)
- Firewall should permit inbound TCP connection to our public webserver at IP address 1.2.2.3

Stateful Filtering

- Suppose you want to allow inbound connection to a FTP server, but block any attempts to login as “root”. How would you build a stateful packet filter to do that? In particular, what state would it keep, for each connection?

- Background: To log in, the FTP client sends “`USER alice`” then “`PASS theyllneverguessthis`” over a TCP connection to the server.

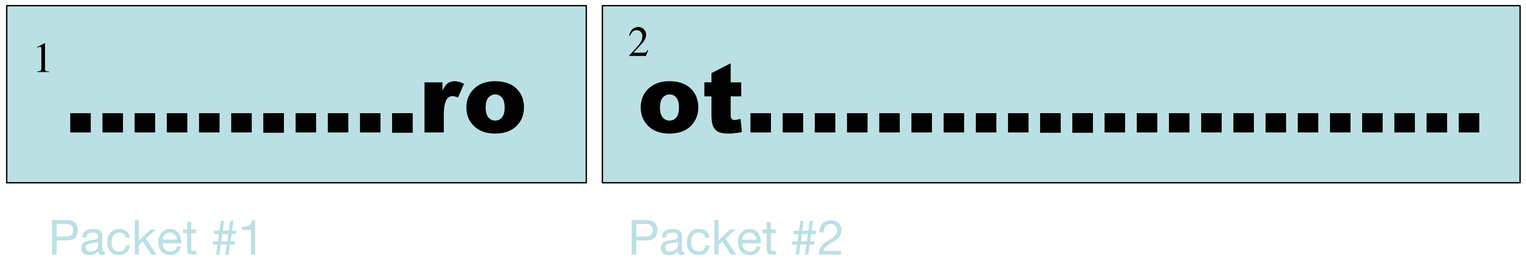
State Kept

- No state – just drop any packet with root in them

- Is it a FTP connection?
- Where in FTP state (e.g. command, what command)
- Src ip addr, dst ip addr, src port, dst port
- Inbound/outbound connection
- Keep piece of login command until it's completed – only first 5 bytes of username

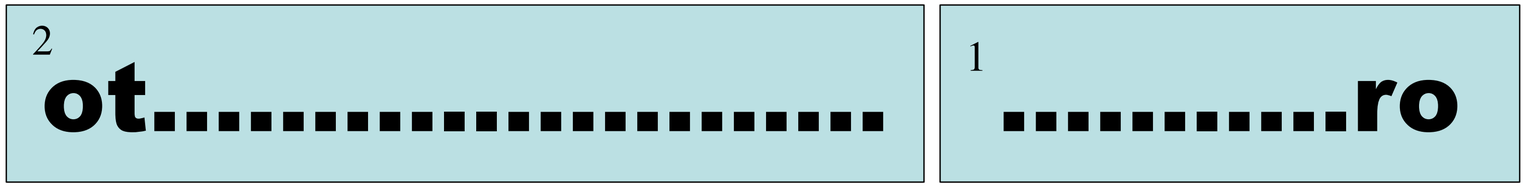
Beware!

- Sender might be malicious and trying to sneak through firewall
- “root” might span packet boundaries

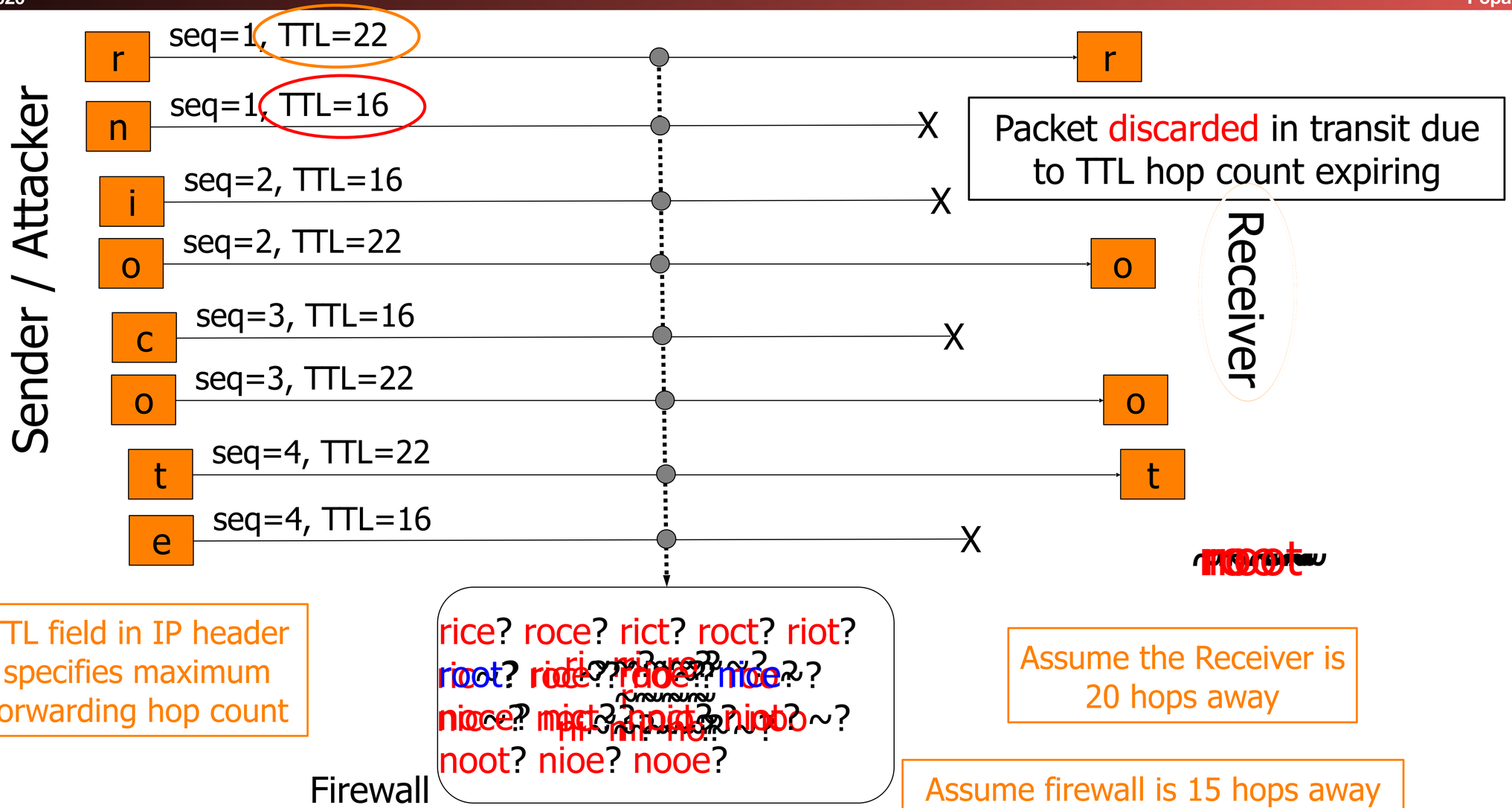


Beware!

- Packets might be re-ordered



Beware!

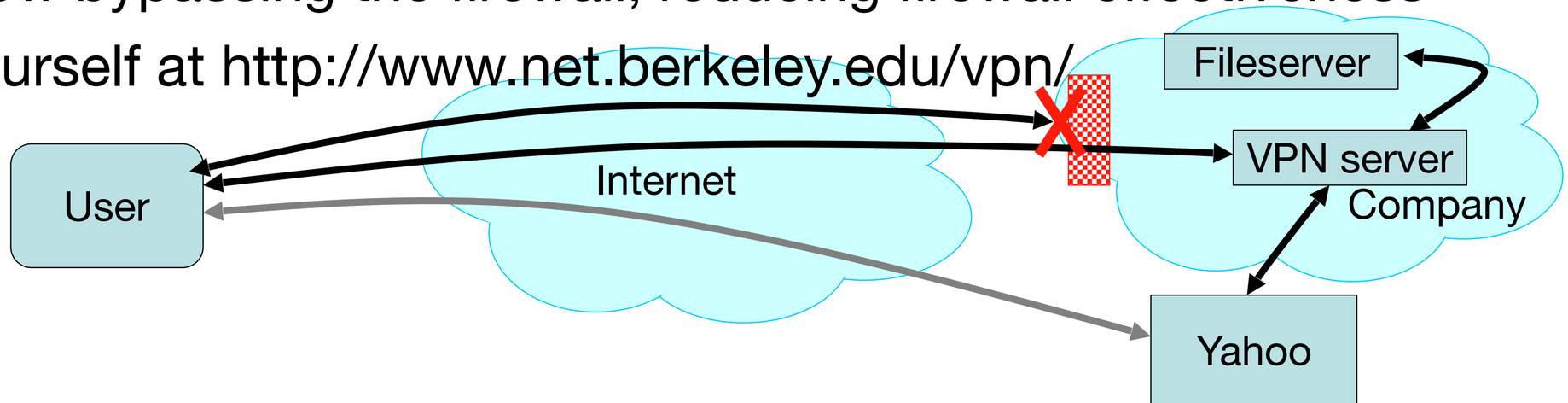

Sender / Attacker
r
seq=1, TTL=22
n
seq=1, TTL=16
i
seq=2, TTL=16
o
seq=2, TTL=22
c
seq=3, TTL=16
o
seq=3, TTL=22
t
seq=4, TTL=22
e
seq=4, TTL=16
X
X
X
X
r
o
o
t
Receiver
Packet discarded in transit due to TTL hop count expiring
TTL field in IP header specifies maximum forwarding hop count
rice? roce? rict? roct? riot?
noot? nioe? nooe?
Firewall
Assume the Receiver is 20 hops away
Assume firewall is 15 hops away

Other Kinds of Firewalls

- Application-level firewall
  - Firewall acts as a proxy. TCP connection from client to firewall, which then makes a second TCP connection from firewall to server.
  - Eliminates risks of stateful packet filter interpreting packets different from end host.

Secure External Access to Inside Machines

- Often need to provide secure remote access to a network protected by a firewall
  - Remote access, telecommuting, branch offices, …
- Create secure channel (Virtual Private Network, or VPN) to tunnel traffic from outside host/network to inside network
  - May allow bypassing the firewall, reducing firewall effectiveness
  - Try it yourself at http://www.net.berkeley.edu/vpn/



Why Have Firewalls Been Successful?

- Central control – easy administration and update
  - Single point of control: update one config to change security policies
  - Potentially allows rapid response
- Easy to deploy – transparent to end users
  - Easy incremental/total deployment to protect 1000's
- Addresses an important problem
  - Security vulnerabilities in network services are rampant
  - Easier to use firewall than to directly secure code …

Think like an attacker

- Suppose you wanted to attack a company protected by a firewall. What attacks might you try?

- Share your ideas on chat (mark it visible to everyone)

Firewall Disadvantages

- Functionality loss – less connectivity, less risk
  - May reduce network's usefulness
  - Some applications don't work with firewalls
    - Two peer-to-peer users behind different firewalls
- The malicious insider problem
  - Assume insiders are trusted
    - Malicious insider (or anyone gaining control of internal machine) can wreak havoc
- Firewalls establish a security perimeter
  - Like Eskimo Pies: "hard crunchy exterior, soft creamy center"
  - Threat from travelers with laptops, cell phones, …

Lateral Movement

- Common attack: compromise an internal machine, then use that to attack other internal machines
- From there, you can now exploit internal systems directly
  - Bypassing the primary firewall
- That is the shortcoming of firewalls: A ***single*** breach of the perimeter by an attacker and you can no longer make ***any*** assertions about subsequent internal state

Takeaways on Firewalls

- Firewalls: Reference monitors and access control all over again, but at the network level
- Attack surface reduction
- Centralized control