Firewalls

CS 161: Computer Security
Prof. David Wagner

March 7, 2013
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
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
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)
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*
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, drop it.
Example Rule

allow tcp connection *:*/*in --> 3.1.1.2:80/out

• 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 /in indicates the network interface.
Example Ruleset

allow tcp connection *:*/in -> *:*/out
allow tcp connection *:*/out -> 1.2.2.3:80/in

• 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
Other Kinds of Firewalls

• Stateless packet filter
  – No state in the packet filter. Rules specify whether to drop packet, without history.
  – Problem: requires hacks to handle TCP connections (e.g., an inbound packet is OK if it is associated with a TCP connection initiated by an inside host to an outside host).

• 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.
  – Only modest benefits over stateful packet filter.
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
  - Provides Authentication, Confidentiality, Integrity
  - However, also raises *perimeter issues*
    (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 …
Firewall Disadvantages?

Discussion question:

What are the limitations of firewalls?
Why have firewalls become less effective over time?

Discuss with a partner.
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, …
Takeaways on Firewalls

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

• Prevent, Detect and respond, Deter, Tolerate
• Detection might enable…
  – Recovery: if I know my machine is infected, I can recover (nuke it from orbit and re-install)
  – Risk management: if I can measure prevalence of different attacks, I can prioritize spending on different defenses wisely
  – Deterrence: if I can detect the attack and attribute the source, maybe we can punish/prosecute the attacker – deterring others in the future
• If we can detect an attack, why not just block it when you detect it?
  – False alarms: detector might occasionally make false positives, and it’d be costly to block legitimate activity
  – After-the-fact response: might be easier to detect attack later than to detect attack in real time
The Problem of Detecting Attacks

- Given a choice, we’d like our systems to be airtight-secure
- But often we don’t have that choice
  - #1 reason why not: cost (in different dimensions)
- A (messy) alternative: detect misuse rather than build a system that can’t be misused
  - Upon detection: clean up damage, maybe block incipient “intrusion”
  - Note: can be prudent for us to do this even if we think system is solid – defense in depth
  - Note: “misuse” might be about policy rather than security
    - Example: your own employees shouldn’t be using file-sharing apps
- Problem space:
  - Lacks principles
  - Has many dimensions (where to monitor, how to look for problems, how much accuracy required, what can attackers due to elude us)
  - Is messy and in practice also very useful
Example Scenario

• Suppose you’ve been hired to provide computer security for FooCorp. They offer web-based services via backend programs invoked via URLs:
  – \texttt{http://foocorp.com/amazeme.exe?profile=info/luser.txt}
  – Script makes sure that “profile” argument is a relative filename
Structure of FooCorp Web Services

Internet

FooCorp’s border router

FooCorp Servers

Remote client

1. GET /amazeme.exe?profile=xxx
2. GET /amazeme.exe?profile=xxx
3. GET /amazeme.exe?profile=xxx
4. amazeme.exe?profile=xxx
5. bin/amazeme -p xxx
Structure of FooCorp Web Services

1. Remote client
   2. Internet
   3. FooCorp’s border router
   4. FooCorp Servers
   5. bin/amazeme -p xxx
   6. Output of bin/amazeme sent back
   7. 200 OK
      Output of bin/amazeme
   8. 200 OK
      Output of bin/amazeme
   9. 200 OK
      Output of bin/amazeme
   10. Browser renders output
Example Scenario

• Suppose you’ve been hired to provide computer security for FooCorp. They offer web-based services via backend programs invoked via URLs:
  – Script makes sure that “profile” argument is a relative filename

• Due to installed base issues, you can’t alter backend components like amazeme.exe

• One of the zillion of attacks you’re worried about is information leakage via directory traversal:
  – E.g. GET /amazeme.exe?profile=../../../../../etc/passwd
Problem with accessing the AmazeMe FooCorp service

Error parsing profile: ../../..../etc/passwd
Can't find foreground/background color preferences in:

root:fo8bXK3L6xI:0:0:Administrator:/bin/sh
flash:pR.33HwJa2c:51:51:Flash User:/flash:/bin/false
nobody:*:99:99:Nobody: /
juser:IT9q23cjwVs:500:503:Jerome L. User:/home/jlusr:/bin/tcsh
hefalump:bKKdz92sk1b:501:503:Mr. Hef:/home/hef:/bin/bash
backdoor:9aBz331dDe1:0:0:Emergency Access:/bin/sh
ncsd:$1GnYOsA552:505:505:NSCD Daemon:/ncsd:/sbin/nologin

Please correct the profile entries and resubmit.

Thank you for using FooCorp.

Helpful error message returns contents of profile that appeared mis-formed, revealing the raw password file
Example Scenario

- Suppose you’ve been hired to provide computer security for FooCorp. They offer web-based services via backend programs invoked via URLs:
  - Script makes sure that “profile” argument is a relative filename
- Due to installed base issues, you can’t alter backend components like amazeme.exe
- One of the zillion of attacks you’re worried about is information leakage via directory traversal:
  - E.g. GET /amazeme.exe?profile=../../../../../etc/passwd
- What different approaches could detect this attack?
Extra Materials
Subverting Firewalls

• Along with possible bugs, packet filters have a fundamentally **limited semantic model**
  – They lack a full understanding of the meaning of the traffic they carry
    o In part because operate only at layers 3 & 4; not 7

• How can a **local user** who wants to get around their site’s firewall exploit this?
  – *(Note: we’re not talking about how an external attacker can escape a firewall’s restrictions)*

• One method of subversion: **abuse ports**
  – Who says that e.g. port 22/tcp = SSH?
    o Why couldn’t it be say Skype or BitTorrent?
    o Just requires that client & server agree on app protocol
Hiding on Other Ports

- Method #1: use port allocated to another service (how can this be detected?)

- Method #2: tunneling
  - Encapsulate one protocol inside another
  - Receiver of “outer” protocol *decapsulates* interior tunneled protocol to recover it
  - Pretty much *any* protocol can be tunneled over another (with enough effort)

- E.g., tunneling IP over SMTP
  - Just need a way to code an IP datagram as an email message (either mail body or just headers)
Example: Tunneling IP over Email

From: doesnt-matter@bogus.com
To: my-buddy@tunnel-decapsulators.R.us
Subject: Here’s my IP datagram

IP-header-version: 4
IP-header-len: 5
IP-ID: 11234
IP-src: 1.2.3.4
IP-dst: 5.6.7.8
IP-payload: 0xa144bf2c0102…

Remote email server receives this legal email, builds an IP packet corresponding to description in email body … … and injects it into the network

How can a firewall detect this??

This operator of this email server has chosen to cooperate with the email sender to help them tunnel
Tunneling, cont.

- E.g., IP-over-ICMP:
  - Embed IP datagram as the payload of a “ping” packet

- E.g., Skype-over-HTTP:
  - Encode Skype messages in URL of requests and header fields of replies

- Note #1: to tunnel, the sender and receiver must both cooperate (so it’s not useful for initial attacks)

- Note #2: tunneling has many legitimate uses too
  - E.g., Virtual Private Networks (VPNs)
    - Make a remote machine look like it’s local to its home network
    - Tunnel encrypts traffic for privacy & to prevent meddling
Application-level Firewall

• Can more directly control applications by requiring them to go through a proxy for external access
  – Proxy doesn’t simply forward, but acts as an application-level middleman

• Example: SSH gateway
  – Require all SSH in/out of site to go through gateway
  – Gateway logs authentication, inspects decrypted text
  – Site’s firewall configured to prohibit any other SSH access
SSH Gateway Example

- **host-to-gateway SSH session**
- **gateway-to-remote host SSH session**
- 1.3.5.7

Firewall

- **allow**
  - <port=22, host=1.3.5.7>
- **drop**
  - <port=22>
Application-level Firewall

- Can more directly control applications by requiring them to go through a proxy for external access
  - Proxy doesn’t simply forward, but acts as an application-level middleman
- Example: SSH gateway
  - Require all SSH in/out of site to go through gateway
  - Gateway logs authentication, inspects decrypted text
  - Site’s firewall configured to prohibit any other SSH access
- Provides a powerful degree of monitoring/control
- Costs?
  - Need to run extra server(s) per app (possible bottleneck)
  - Each server requires careful hardening
FW Disadvantages, con’t

• “Malicious” applications
  – Previous properties combine in a very nasty way: app protocol blocked by users’ firewalls

• What to do?
  – Tunnel app’s connections over HTTP or SMTP
  – Web is killer app, so most firewalls allow it
  – Now firewall can’t distinguish real/app traffic
  – Insiders trusted ⇒ their apps trusted ⇒ firewall can’t protect against malicious apps
  – More and more traffic goes over port 25/80/…
    • Firewalls have less visibility into traffic
    • Firewalls become less effective
Security Principle: *Reference Monitors*

- Firewalls embody useful **principles** that are applicable elsewhere in computer security
  - Optimized for enforcing particular kind of **access control policy**
  - **Chokepoint** notion makes enforcement possible
- A **key** conceptual approach to access control: **reference monitor**
  - Examines **every** request to access a controlled resource (an **object**) and determines whether to allow request

```
Subject  Request  Reference Monitor  Object
```
Reference Monitor Security Properties

- **Always invoked**
  - *Complete mediation* property: all security-relevant operations must be mediated by RM
  - RM should be invoked on every operation controlled by access control policy
- **Tamper-resistant**
  - Maintain RM *integrity* (no code/state tampering)
- **Verifiable**
  - Can *verify* RM operation (correctly enforces desired access control policy)
    - Requires extremely *simple* RM
    - We find we *can’t verify* correctness for systems with any appreciable degree of *complexity*
Considering Firewalls as Reference Monitors

• Always invoked?
  – Place Packet Filter as an in-path element on chokepoint link for all internal-external communications
  – Packets only forwarded across link if firewall explicitly decides to do so after inspection
Potential Problems?

• What if a user hooks up an unsecured wireless access point to their internal machine?
• Anyone who drives by with wireless-enabled laptop can gain access to internal network – Bypasses packet filter!

• To use a firewall safely, must ensure we’ve covered all links between internal and external networks with firewalls – Set of links known as the security perimeter
RM Property: Tamper-Resistant

• Will this hold?

• Do not allow management access to firewall other than from specific hosts
  – I.e., firewall itself needs firewalling

• Protect firewall’s physical security

• Must also secure storage & propagation of configuration data
RM Property: Verifiable

• Will this hold?

• Current practice:
  – Packet filter software too complex for feasible systematic verification …
  – … and rulesets with 1,000s (!) of rules

• Result:
  – *Bugs* that allowed attackers to defeat intended security policy by sending unexpected packets that packet filter doesn’t handle as desired
Stateless Packet Filters

• Basic kind of firewall: *stateless packet filter*
  – Router with list of *access control rules*
  – Router checks each received packet against security rules to decide to forward or drop it
  – Each rule specifies which packets it applies to based on a packet’s header fields (*stateless*)
    • Specify source and destination IP addresses, port numbers, and protocol names, or *wild cards*
    • Each rule specifies the *action* for matching packets: *ALLOW* or *DROP* (aka DENY)
      \[
      \texttt{<ACTION> <PROTO> <SRC:PORT> \rightarrow <DST:PORT>}
      \]
  – First listed rule has *precedence*
Examples of Packet Filter Rules

allow tcp 4.5.5.4:1025 -> 3.1.1.2:80

• States that the firewall should **permit** any TCP packet that’s:
  – from Internet address 4.5.5.4 **and**
  – using a source port of 1025 **and**
  – destined to port 80 of Internet address 3.1.1.2

deny tcp 4.5.5.4:* -> 3.1.1.2:80

• States that the firewall should **drop** any TCP packet like the above, regardless of source port
Examples of Packet Filter Rules

- **deny** tcp 4.5.5.4:* 
  > 3.1.1.2:80

- **allow** tcp 4.5.5.4:1025 
  > 3.1.1.2:80

  - **In this order**, the rules won’t allow any TCP packets from 4.5.5.4 to port 80 of 3.1.1.2

- **allow** tcp 4.5.5.4:1025 
  > 3.1.1.2:80

- **deny** tcp 4.5.5.4:* 
  > 3.1.1.2:80

  - **In this order**, the rules allow TCP packets from 4.5.5.4 to port 80 of 3.1.1.2 **only** if they come from source port 1025
Expressing Policy with Rulesets

• Goal: prevent *external access* to Windows SMB (TCP port 445)
  – Except for one special external host, 8.4.4.1

• Ruleset:

```plaintext
allow tcp 8.4.4.1:* -> *:445
drop tcp *:* -> *:445
allow * *:* -> *:*
```

• Problems?
  – No notion of *inbound* vs *outbound* connections
    • Drops outbound SMB connections from inside users
  – (This is a *default-allow* policy!)
Expressing Policy with Rulesets

- Want to allow:
  - Inbound mail connections to our mail server (1.2.3.4:25)
  - All outbound connections from our network, 1.2.3.0/24
    - 1.2.3/24 = “any address for which the top 24 bits match 1.2.3.0”
    - So it ranges from 1.2.3.0, 1.2.3.1, …, 1.2.3.255
  - Nothing else

- Consider this ruleset:
  ```
  allow tcp *:* -> 1.2.3.4:25
  allow tcp 1.2.3.0/24:* -> *:*
  drop * *:* -> *:* 
  ```

- This policy doesn't work …
  - TCP connections are bidirectional
  - 3-way handshake: client sends SYN, receives SYN+ACK, sends ACK
    - Followed by either/both sides sending DATA (w/ ACK bit set)
Problem: Outbound Connections Fail

1. allow tcp *:* -> 1.2.3.4:25
2. allow tcp 1.2.3.0/24:* -> *:*
3. drop * *:*:* -> *:*:

• Inside host opens TCP connection to port 80 on external machine:
  – Initial SYN packet passed through by rule 2
  – SYN+ACK packet coming back is dropped
  • Fails rule 1 (not destined for port 25)
  • Fails rule 2 (source not inside host)
  • Matches rule 3 ⇒ DROP
Problem: Outbound Connections Fail

1. allow tcp *:* -> 1.2.3.4:25
2. allow tcp 1.2.3.0/24:* -> *:*  
3. drop * *:* -> *:* 

• Fix?
  – In general, we need to distinguish between 2 kinds of inbound packets
    • Allow inbound packets associated with an outbound connection
    • Restrict inbound packets associated with an inbound connection
  – How do we tell them apart?
    • Approach #1: remember previous outbound connections
      – Requires state :-(
    • Approach #2: leverage details of how TCP works …
Inbound vs. Outbound Connections

• Key TCP feature: ACK bit set on all packets except first
  – **Plus:** TCP receiver disregards packets with ACK set if they don’t belong to an existing connection

• Solution ruleset:

  1. allow tcp *:* -> 1.2.3.4:25
  2. allow tcp 1.2.3.0/24:* -> *:*  
  3. allow tcp *:* -> 1.2.3.0/24:* only if ACK bit set
  4. drop * *:* -> *:*  

  – Rules 1 and 2 allow traffic in either direction for inbound connections to port 25 on machine 1.2.3.4
  – Rules 2 and 3 allow outbound connections to any port
How This Ruleset Protects

1. allow tcp *:* -> 1.2.3.4:25
2. allow tcp 1.2.3.0/24:* -> *:* only if ACK bit set
3. allow tcp *:* -> 1.2.3.0/24:* only if ACK bit set
4. drop * *:* -> *:* 

• Suppose external attacker tries to exploit vulnerability in SMB (TCP port 445):
  = Attempts to open an inbound TCP connection to internal SMB server

• Attempt #1: Sends SYN packet to server
  – Packet lacks ACK bit ⇒ no match to Rules 1-3, dropped by Rule 4

• Attempt #2: Sends SYN+ACK packet to server
  – Firewall permits the packet due to Rule 3
  – But then dropped by server’s TCP stack (since ACK bit set, but isn’t part of existing connection)