Basic Definitions

- **Firewall**: A component or set of components that restricts services between two networks
  - often the two networks are the Internet and an “internal” network
- **Bastion host**: A computer system that must be highly secured because it is vulnerable to attack
  - usually is exposed to the Internet and is the main point of contact for remote users of the internal network
- **Dual-homed host**: A general-purpose computer with two or more network interfaces
- **Network address translation (NAT)**: Procedure by which a router alters source or destination addresses in packets
  - not really a security technique, but can augment security and is often performed at a firewall
Basic Definitions (cont.)

- Packet filtering: Selectively passing or blocking packets, usually while routing them from one network to another
  - Can occur in a router, bridge, or host
  - Also called “screening”

- Perimeter network: A network added between an external network and a protected (internal) network, in order to provide an additional level of security
  - Also called a “demilitarized zone” (DMZ)

- Proxy: A program that interacts with external servers on behalf of internal clients

- Virtual private network: Packets that are internal to a private network pass across a public network, without this being obvious to hosts on the internal network
Data Available to a Packet Filter

- **Header data**
  - IP source and destination addresses
  - Transport protocol (TCP, UDP, or ICMP)
  - TCP/UDP source and destination ports
  - ICMP message type
  - Packet size

- **Packet filter can look further into the packet**
  - e.g., the URL being requested

- **Whether the packet is well-formed**
  - is packet the size it claims to be?
  - is it formatted properly for its destination port?

Data Available to Packet Filter (cont.)

- **The interface the packet arrived on**

- **The interface the packet would leave on**

- **And if the filter keeps state ...**
  - Whether this packet appears to be a response to another packet it has recently passed
  - How many packets have been seen recently to or from the same host
  - Whether this packet is identical to a recently sent packet
  - If this packet is part of a larger packet that was fragmented
Actions Available to the Packet Filter

- Send the packet toward its intended destination
- Drop the packet, without notifying the sender
- Reject the packet, with notification to the sender
  - e.g., an ICMP “destination unreachable” packet
- Log information about the packet
- Set off an alarm
- Modify the packet (e.g., NAT)
- Send the packet to other than its intended destination
  - e.g., a proxy or to enforce load balancing
- Modify the filtering rules
  - e.g., accept replies to a UDP packet, or stop all traffic from a host that has sent malformed packets

Examples of Packet Filtering

- Block all incoming connections from systems outside the internal network, except for SMTP connections
- Block all connections to or from systems you distrust
- Block or log all connections to specified domains
  - particularly common for pornographic sites
- Allow electronic mail and FTP, but disallow X11, rsh, rcp, ...
Pros and Cons of Packet Filtering

Advantages:
- One screening router can protect an entire network
- Simple packet filtering can be extremely efficient

Disadvantages:
- Hard to configure and test
- Is susceptible to “failing open”
- Can be slow (even if simple)
  - filtering is incompatible with certain optimizations
- Cannot implement many useful policies
  - does not have access to user who initiated a packet
  - packets say what port they’re for, but not what application will receive them

Proxies

- Special servers that accept client requests to servers and perform them on client’s behalf
  - generally transparent to client user and server
- Effective only when direct client-server interactions prevented
  - otherwise, proxy will be bypassed
Types of Proxies

- Usually used to control outbound connections, but can also be used to control inbound connections
  - controlling inbound connections often called “reverse proxying”

- Example proxy: ftp proxy that permits internal users to import files but prohibits them from exporting files

- Example reverse proxy: balancing incoming requests among multiple servers

Advantages of Proxies

- Can be good at logging
  - e.g., log only ftp commands, not all data transferred

- Can cache content
  - decreases response latency for client

- Can filter more intelligently than a packet filter
  - filter viruses, active content (Java, Javascript), etc.

- Can perform user-level authentication
  - take actions based on which user is issuing requests

- Can protect clients from malformed IP packets
  - generates new IP packets to clients
Disadvantages of Proxies

- Proxy availability lags behind introduction of new services
- Typically a new proxy is required for each service
  - though some can be run through generic proxies
- Usually require modifications to client applications

Network Address Translation (NAT)

- Can dynamically allocate external address and port for each connection initiated by an internal host
- Not only (or even primarily) a security technology
  - mainly used to multiplex numerous IP addresses over a few
Security Advantages of NAT

- **Enforces firewall’s control over outbound connections**
  - if a connection bypasses the firewall, it won’t work because its address is not valid on the external network

- **Temporally restricts incoming traffic**
  - dynamic translation allows only packets that are part of a current interaction initiated from the inside
  - once translation goes away, address that the attacker knows is no longer usable

- **Helps to conceal internal network configuration**
  - how many internal hosts there are, for example

Disadvantages of NAT

- **Dynamic allocation requires state information that is not always available**
  - How long should the translator keep a translation for the external address inserted into an outbound UDP packet?

- **Embedded IP addresses are a problem for NAT**
  - NAT systems normally translate the header, but some protocols bury IP addresses elsewhere

- **NAT can break authentication**
  - NAT is incompatible with IPSec transport mode
  - Integrity-protected, embedded IP addresses are hopeless

- **Logging after translation yields confusing logs**
  - “Reconstructing” log requires precise clock synchronization and time correlation
Virtual Private Networks (VPNs)

- Cryptographic techniques applied to traffic between two distant networks or between end host and network
  - IPSec the most widely used cryptographic protection, most commonly in tunnel mode
- Where to end tunnel?

<table>
<thead>
<tr>
<th>In internal network</th>
<th>In external network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall</td>
<td>Firewall</td>
</tr>
<tr>
<td>Internal network</td>
<td>External network</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>firewall can’t do its job</td>
<td>traffic exposed too soon</td>
</tr>
</tbody>
</table>

Pros and Cons of VPNs

Advantages:
- Provide strong confidentiality and authenticity of traffic
  - channel authenticated only to granularity of tunnel endpoint
- Enables remote use of protocols that would be difficult to secure any other way

Disadvantages:
- VPNs involve dangerous network connections
  - particularly from mobile devices, which may come under attack
  - ideally, client VPN software disables other uses of client network interface while VPN is in use
- VPNs extend the perimeter that must be secured
Single Box Architectures

- Very high performance
  - a favorite of ISPs
- Primarily packet filtering

- Very high degree of control
  - everything is proxied, or users must log into it to access outside
- Challenge: securing the firewall itself and keeping it alive
  - lots ends up running there

Screened Host Architectures

- Bastion host attached only to internal net
  - the only computer that external network can connect to
  - only some types of connections allowed
- Security mainly provided by packet filter
- Packet filter can selectively allow some connections from other internal hosts
- Both bastion host and router are “single points of failure”
Screened Subnet Architectures

- Two screening routers
- Better protection against bastion host compromise
- No single point of failure

Bastion host

Perimeter network

Motivation for Perimeter Network

- Many networking technologies permit any machine on the network to see all traffic on the network
  - Ethernet, Token ring, FDDI

- All traffic on the perimeter network should be
  - to/from External network
  - to/from bastion host

- Thus, no entirely internal traffic should be exposed to an attacker who compromises the bastion host
Bastion Host

- **Main point of contact for incoming connections from external network**
  - For incoming email (SMTP) sessions to deliver electronic mail to the site
  - For incoming FTP connections to site’s anonymous FTP server
  - For incoming DNS queries about the site

- **Outbound services handled one of two ways**
  - Routers set up to allow direct internal-to-external connections
  - Proxy runs on bastion host
    - Internal filter permits internal clients to connect to proxy server on bastion host

Interior Router

- **Sometimes called the “choke router”**

- **Performs most of the packet filtering for your firewall**
  - Permits some internal hosts to connect to external servers
    - Possible examples are HTTP and telnet
  - For other services, internal hosts forced connect to proxies on bastion host

- **Should permit connections only to selected internal hosts**
  - And usually only from the bastion host
Exterior Router

- Sometimes called the “access router”
- Filtering rules
  - Duplicate many of the filtering rules on the internal router
  - Permit outbound connections from bastion host proxies

Two main jobs
- Filters incoming packets with forged source addresses
  - Prevents outsiders from forging packets that
    - appear to be from hosts on the perimeter network
    - appear to be from hosts on the internal network
- Filters outgoing packets with forged source addresses
  - An important part of being a good “network citizen”

Split-Screened Subnet Architecture

- Routers protect from
  - address forgeries
  - protection failures of dual homed host
Independent Screened Subnets

- Provides redundancy
  - No single point of failure for Internet connectivity

- Greater privacy, e.g.,
  - External network 1 = Internet
  - External network 2 = Supplier network

- Run inbound services across one, outbound across the other
  - Both are easier to secure if separated
Example ISP Firewall

Variations: Merge Interior & Exterior Routers

- Requires highly capable screening router
  - Must support inbound and outbound filters on each interface
- Creates a single point of failure (screening router)
  - Like screened host architecture
  - But routers are easier to protect than hosts
**Variations: Merge Bastion Host & Exterior Router**

- May expose bastion host further
- If bastion host is dual-homed, then may perform worse

**Dangerous: Multiple Interior Routers**

- Risk that internal traffic will be routed over perimeter network
  - Compromise of bastion host will permit internal traffic to be snooped
Multiple Interior Routers (cont.)

- Though dangerous, it provides redundancy and increased performance ... but ...

- If redundancy is motivating factor, then independent screened subnets are better

- If performance is motivating factor, then either
  - A lot of traffic going to perimeter network is not then going to external network
  - This probably means a misconfiguration
  - The exterior router is much faster than your interior router
  - Better to upgrade your interior router than buy another

Another argument for multiple interior routers is to support multiple internal networks that should be protected from each other.

A better alternative is to give them separate interfaces on one router.
### Multiple Interior Routers (cont.)

- If there are too many internal networks for one router, set up a backbone

![Diagram of network architecture including external network, perimeter network, backbone, internal network 1, and internal network 2.](image)

### Types of Packet Filtering: By Address

- **Simplest form of filtering**
- **Restricts flow based on source and/or destination addresses**
  - Does not consider the protocol involved
- **Mainly used to prevent insertion of packets with forged source addresses**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Direction</th>
<th>Source address</th>
<th>Destination address</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Inbound</td>
<td>Internal</td>
<td>Any</td>
<td>Deny</td>
</tr>
</tbody>
</table>

- **Notation**
  - “Inbound” is relative to internal network
  - “Internal” and “Any” are abbreviations for IP address ranges
  - Rules applied in sequential order until match is found
Types of Packet Filtering: By Service

- Filtering by service is more common, but also more complex
- As an example, consider filtering telnet
- Outbound telnet
  - Characteristics of outgoing packets
    - Telnet is a TCP-based service, so the IP packet type is TCP
    - The TCP destination port is 23
    - The TCP source port number is a number \( y > 1023 \)
    - First outgoing packet will not have the ACK bit set; others will
  - Characteristics of incoming packets
    - TCP source port is 23
    - TCP destination port is \( y \)
    - Has the ACK bit set

Packet Filtering by Service (cont.)

- Example filtering rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Direction</th>
<th>Source address</th>
<th>Destination address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Destination port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Out</td>
<td>Internal</td>
<td>Any</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>23</td>
<td>Either</td>
<td>Permit</td>
</tr>
<tr>
<td>B</td>
<td>In</td>
<td>Any</td>
<td>Internal</td>
<td>TCP</td>
<td>23</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
<tr>
<td>C</td>
<td>Either</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Either</td>
<td>Deny</td>
<td></td>
</tr>
</tbody>
</table>

- Does not enforce telnet characteristics exactly
- In fact, permits some seemingly dangerous communication
  - Example: Inbound packets with source port 23 to any port > 1023 will be accepted, if the ACK bit is set
  - Only way to fix this is by keeping some state, or using a proxy
Effect of Order on Filtering

Consider the following example

You’re in a corporation working on a project with a university
Corporate network is 172.16 (i.e., 172.16.0.0 to 172.16.255.255)
University owns network 10 (i.e., 10.0.0.0 to 10.255.255.255)
You’re going to link these networks together using a packet filter
You want to disallow all Internet access over this link
Project uses the 172.16.6 subnet
University’s 10.1.99 subnet has lots of hostile activity

Suppose we try the following filtering rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Source address</th>
<th>Destination address</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10.*.<em>.</em></td>
<td>172.16.6.*</td>
<td>Permit</td>
</tr>
<tr>
<td>B</td>
<td>10.1.99.*</td>
<td>172.16.<em>.</em></td>
<td>Deny</td>
</tr>
<tr>
<td>C</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
</tbody>
</table>

Effect of Order on Filtering (cont.)

Consider several example packets, assuming rules are applied in order ABC

<table>
<thead>
<tr>
<th>Packet</th>
<th>Source address</th>
<th>Destination address</th>
<th>Desired action</th>
<th>Actual action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.1.99.1</td>
<td>172.16.1.1</td>
<td>Deny</td>
<td>Deny (B)</td>
</tr>
<tr>
<td>2</td>
<td>10.1.99.1</td>
<td>172.16.6.1</td>
<td>Permit</td>
<td>Permit (A)</td>
</tr>
<tr>
<td>3</td>
<td>10.1.1.1</td>
<td>172.16.6.1</td>
<td>Permit</td>
<td>Permit (A)</td>
</tr>
<tr>
<td>4</td>
<td>10.1.1.1</td>
<td>172.16.1.1</td>
<td>Deny</td>
<td>Deny (C)</td>
</tr>
<tr>
<td>5</td>
<td>192.168.3.4</td>
<td>172.16.1.1</td>
<td>Deny</td>
<td>Deny (C)</td>
</tr>
<tr>
<td>6</td>
<td>192.168.3.4</td>
<td>172.16.6.1</td>
<td>Deny</td>
<td>Deny (C)</td>
</tr>
</tbody>
</table>
Effect of Order on Filtering (cont.)

- Now suppose the firewall reorders the rules by the number of significant bits in the source address field, resulting in BAC
  - More specific rules are applied first

<table>
<thead>
<tr>
<th>Packet</th>
<th>Source address</th>
<th>Destination address</th>
<th>Desired action</th>
<th>Actual action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.1.99.1</td>
<td>172.16.1.1</td>
<td>Deny</td>
<td>Deny (B)</td>
</tr>
<tr>
<td>2</td>
<td>10.1.99.1</td>
<td>172.16.6.1</td>
<td>Permit</td>
<td>Deny (B)</td>
</tr>
<tr>
<td>3</td>
<td>10.1.1.1</td>
<td>172.16.6.1</td>
<td>Permit</td>
<td>Permit (A)</td>
</tr>
<tr>
<td>4</td>
<td>10.1.1.1</td>
<td>172.16.1.1</td>
<td>Deny</td>
<td>Deny (C)</td>
</tr>
<tr>
<td>5</td>
<td>192.168.3.4</td>
<td>172.16.1.1</td>
<td>Deny</td>
<td>Deny (C)</td>
</tr>
<tr>
<td>6</td>
<td>192.168.3.4</td>
<td>172.16.6.1</td>
<td>Deny</td>
<td>Deny (C)</td>
</tr>
</tbody>
</table>

- Turns out that B is redundant, anyway

Proxying

- Redirection of client request to proxy server usually happens by one of the following four approaches
  - Proxy-aware client application software
  - Proxy-aware client operating system
  - Proxy-aware user procedures (and so the illusion diminishes)
  - Proxy-aware router redirects client request
How Proxying Works

- **Proxy-aware client application software**
  - Not available for all applications and platforms
  - Generally requires user configuration, and so may not be transparent

- **Proxy-aware client operating system**
  - When the application tries to make a connection, the O/S invokes the proxy server instead
  - Easiest to do this using a dynamically linked library that handles networking calls; otherwise, network drivers need to be modified
  - Is fairly fragile; problems arise with
    - Statically linked software
    - Software that provides its own dynamically linked libraries for networking functions
    - Protocols that use embedded port numbers or IP addresses
    - Software that manipulates connections at a low level

How Proxying Works (cont.)

- **Proxy-aware user procedures**
  - User tells (unmodified) client to connect to proxy server, and then tells proxy server which host to connect to
  - Example: To retrieve a file from anonymous ftp server ftp.foo.com:
    - User, using any ftp client, connects to proxy server, instead of ftp.foo.com
    - At username prompt, user specifies both account name and real server she wants to connect to: anonymous@ftp.foo.com
  - Of course, this is no longer transparent to user

- **Proxy-aware router**
  - Also called “hybrid proxying” or “transparent proxying”
  - Most transparent of the options: client is unchanged
  - Also difficult to administer, since it inherits disadvantages of both packet filtering and proxying
Types of Proxy Servers

- "Dedicated" or "Application-level"
  - Understands and interprets the commands in the protocol it proxies
  - Can do intelligent processing
    - Selectively filter or log application-specific commands
    - Caching, e.g., in an HTTP proxy

- "Generic" or "Circuit-level"
  - Roughly equivalent to a packet filter; does not interpret protocol-specific commands or data
  - Does not work for protocols that embed ports or IP addresses in application payload (e.g., FTP)
  - Automatically protect against malformed packet headers and packet fragmentation problems

An Example Firewall

Assumptions

- Screened subnet architecture
- There are hosts on the internal network that fulfill roles of
  - Mail server
  - Usenet news server
  - DNS server
  - Clients for various Internet services
- Internal users are assumed trustworthy
  - a simplifying assumption for this example, but not a good idea
- All hosts use properly assigned and routed IP addresses
- Separate network numbers for perimeter and internal nets
An Example Firewall: HTTP and HTTPS

- **Incoming HTTP(S):** Web server on bastion host
- **Outgoing HTTP(S):** Two options
  - Packet filtering
    - Allow internal hosts to create connections to external hosts’ port 80, port 443, and any port above 1023
    - Internal hosts can access any port above 1023 with no help from the firewall
  - Proxy server
    - Standard web browsers have built-in support for proxy access
    - Supports HTTP(S) access to any port
    - Can provide caching
  - Let’s assume a proxy server here

An Example Firewall: SMTP

- **Underlying thinking**
  - Connection from bastion host to arbitrary internal host is dangerous
  - Connection from arbitrary external hosts to internal host is dangerous

- **Incoming SMTP**
  - All incoming mail goes to SMTP server on bastion host
    - Achieved using DNS MX records
    - Bastion host passes all incoming mail to single secured internal SMTP server

- **Outgoing SMTP**
  - All internal hosts direct mail to internal SMTP server
An Example Firewall: Telnet

- **Incoming telnet: Disallow**

- **Outgoing telnet: Two options**
  - Proxy server
    - Would be needed if users were untrusted
      - proxy authenticates and monitors them
      - not the case here
    - Proxy server imposes modified clients or user procedures
  - Packet filtering
    - Easier alternative; let’s choose this

An Example Firewall: SSH

- **Permit remote access via SSH (safer than telnet)**

- **Inbound SSH: Two options**
  - SSH to bastion host, and then login to internal target
    - Bastion host can verify that SSH is coming in
    - Bastion host SSH server can be carefully configured
    - Requires user accounts on bastion host
  - SSH to internal hosts
    - Possibility of SSH servers that do port forwarding, or other servers altogether on SSH port
    - Hopefully this risk will be small, since internal users are trusted
    - We’ll assume SSH to internal hosts

- **Outbound SSH: permit, but warn users of port forwarding**
  - Outgoing SSH can enable incoming attacks if port forwarding is on
An Example Firewall: FTP

- Outbound normal-mode FTP requires incoming connection to an arbitrary port over 1023
  - Allowing this without doing anything else is too permissive
- Outbound FTP: Two (realistic) choices
  - Passive mode via packet filtering, or normal mode via proxies
  - Here, let’s do both
    - Permit passive mode where we can impose clients that support it
      - Note: internal hosts must be able to access any port over 1023, since that may be the data channel 😘
    - Proxy ftp where we can’t, imposing new user procedures
  - Recall that if we wanted to monitor ftp usage, we’d have to proxy exclusively (but we don’t)
- Inbound FTP: Disallow except for anonymous on bastion host

An Example Firewall: NNTP

- Need to have a news server on internal network
  - To support internal newsgroups
  - To support older Unix-based (non-NNTP) news clients, which read news from local files
- News server an administrative pain for bastion host
  - Fail often
  - If anything, put it on a different bastion host, but that’s expensive 😔
- Here, let’s assume we permit direct NNTP transfers from selected external news feeds to our internal news server
  - A somewhat dangerous posture 😘
  - Should use NNTP authentication in this case
An Example Firewall: DNS

- DNS network activities include lookups and zone transfers
  - Zone transfer copies zone from a primary server to a secondary one
  - Zone transfers happen among servers who serve queries for the same zone

- Here, let’s assume we put
  - a secondary server on the bastion host, to serve external queries
  - a primary server on an internal host, to serve internal ones

- Note: no information hiding in secondary server

An Example Firewall: Interior Router

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoof-1</td>
<td>In</td>
<td>Internal</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
<tr>
<td>Spoof-2</td>
<td>Out</td>
<td>External</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
</tbody>
</table>

- Blocks packets with forged IP source addresses

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP-1</td>
<td>Out</td>
<td>Internal</td>
<td>Bastion</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>80</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>HTTP-2</td>
<td>In</td>
<td>Bastion</td>
<td>Internal</td>
<td>TCP</td>
<td>80</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>
**An Example Firewall: Internal Router (cont.)**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telnet-1</td>
<td>Out</td>
<td>Internal</td>
<td>Any</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>23</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>Telnet-2</td>
<td>In</td>
<td>Any</td>
<td>Internal</td>
<td>TCP</td>
<td>23</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Permits outbound telnet connections

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH-1</td>
<td>Out</td>
<td>Internal</td>
<td>Any</td>
<td>TCP</td>
<td>Any</td>
<td>22</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>SSH-2</td>
<td>In</td>
<td>Any</td>
<td>Internal</td>
<td>TCP</td>
<td>22</td>
<td>Any</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Permits outbound ssh connections

  - “Any” instead of “>1023” since some forms of authentication require SSH clients to use ports at or below 1023

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH-3</td>
<td>In</td>
<td>Any</td>
<td>Internal</td>
<td>TCP</td>
<td>Any</td>
<td>22</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>SSH-4</td>
<td>Out</td>
<td>Internal</td>
<td>Any</td>
<td>TCP</td>
<td>22</td>
<td>Any</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Permit incoming SSH connections

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP-1</td>
<td>Out</td>
<td>Internal</td>
<td>Any</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>21</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>FTP-2</td>
<td>In</td>
<td>Any</td>
<td>Internal</td>
<td>TCP</td>
<td>21</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Allow outgoing command-channel connections to FTP servers, for use by passive-mode internal clients

---

Copyright © 2018 by Michael Reiter.  
All rights reserved.
An Example Firewall: Internal Router (cont.)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP-3</td>
<td>Out</td>
<td>Internal</td>
<td>Any</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>&gt;1023</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>FTP-4</td>
<td>In</td>
<td>Any</td>
<td>Internal</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Allow outgoing data-channel connections to FTP servers, for use by passive-mode internal clients
  - A very permissive rule, but required to support passive-mode FTP

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP-5</td>
<td>Out</td>
<td>Internal</td>
<td>Bastion</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>21</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>FTP-6</td>
<td>In</td>
<td>Bastion</td>
<td>Internal</td>
<td>TCP</td>
<td>21</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Allow internal, normal-mode FTP clients to make command-channel connection to FTP proxy on bastion host

- Permits FTP data connections from proxy server on bastion host to normal-mode internal FTP clients

- FTP-7 prevents attacker on bastion host from attacking internal X11 servers via hole created by FTP-8 and FTP-9
  - If other servers are listening on internal ports above 1023, similar rules should be added for them
  - Trying to list things to deny (ala FTP-7) is a losing battle, but the best that can be done in this case
An Example Firewall: Internal Router (cont.)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP-1</td>
<td>Out</td>
<td>Internal SMTP server</td>
<td>Bastion</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>25</td>
<td></td>
<td>Permit</td>
</tr>
<tr>
<td>SMTP-2</td>
<td>In</td>
<td>Bastion</td>
<td>Internal SMTP server</td>
<td>TCP</td>
<td>25</td>
<td>&gt;1023</td>
<td></td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Permit outgoing mail from internal mail server to bastion host

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP-3</td>
<td>In</td>
<td>Bastion</td>
<td>Internal SMTP server</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>25</td>
<td></td>
<td>Permit</td>
</tr>
<tr>
<td>SMTP-4</td>
<td>Out</td>
<td>Internal SMTP server</td>
<td>Bastion</td>
<td>TCP</td>
<td>25</td>
<td>&gt;1023</td>
<td></td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Permit incoming mail from bastion host to internal mail server

An Example Firewall: Internal Router (cont.)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNTP-1</td>
<td>Out</td>
<td>Internal NNTP server</td>
<td>NNTP feed server</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>119</td>
<td></td>
<td>Permit</td>
</tr>
<tr>
<td>NNTP-2</td>
<td>In</td>
<td>NNTP feed server</td>
<td>Internal NNTP server</td>
<td>TCP</td>
<td>119</td>
<td>&gt;1023</td>
<td></td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Allow outgoing news from internal server to service provider

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNTP-3</td>
<td>In</td>
<td>NNTP feed server</td>
<td>Internal NNTP server</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>119</td>
<td></td>
<td>Permit</td>
</tr>
<tr>
<td>NNTP-4</td>
<td>Out</td>
<td>Internal NNTP server</td>
<td>NNTP feed server</td>
<td>TCP</td>
<td>119</td>
<td>&gt;1023</td>
<td></td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Allow incoming news from service provider to internal server
An Example Firewall: Internal Router (cont.)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS-1</td>
<td>Out</td>
<td>Internal DNS server</td>
<td>Bastion</td>
<td>UDP</td>
<td>53</td>
<td>53</td>
<td>Permit</td>
<td></td>
</tr>
<tr>
<td>DNS-2</td>
<td>In</td>
<td>Bastion</td>
<td>Internal DNS server</td>
<td>UDP</td>
<td>53</td>
<td>53</td>
<td>Permit</td>
<td></td>
</tr>
</tbody>
</table>

- Allow UDP-based queries & answers between internal DNS server & bastion DNS server

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS-3</td>
<td>Out</td>
<td>Internal DNS server</td>
<td>Bastion</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>53</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>DNS-4</td>
<td>In</td>
<td>Bastion</td>
<td>Internal DNS server</td>
<td>TCP</td>
<td>53</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Allow TCP-based queries from internal DNS server to bastion DNS server, and their responses

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS-5</td>
<td>In</td>
<td>Bastion</td>
<td>Internal DNS server</td>
<td>TCP</td>
<td>&gt;1023</td>
<td>53</td>
<td>Any</td>
<td>Permit</td>
</tr>
<tr>
<td>DNS-6</td>
<td>Out</td>
<td>Internal DNS server</td>
<td>Bastion</td>
<td>TCP</td>
<td>53</td>
<td>&gt;1023</td>
<td>Yes</td>
<td>Permit</td>
</tr>
</tbody>
</table>

- Allow TCP-based queries from bastion DNS server to internal DNS server, and their responses

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Source address</th>
<th>Dest. Address</th>
<th>Protocol</th>
<th>Source port</th>
<th>Dest. port</th>
<th>ACK set</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default-1</td>
<td>Out</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
<tr>
<td>Default-2</td>
<td>In</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
</tbody>
</table>

- Deny anything not explicitly allowed by the preceding rules