Trusted Information Systems

Internet Firewalls: An Overview

Why a Firewall?

- **To protect one network from another**
  - Keep out unauthorized users
  - Keep in private or sensitive data
- **To comply with existing organizational policies**
- **To audit or log Internet usage**
  - Justify connection through usage statistics
  - Pinpoint bottlenecks
- **To act as a central point of contact for an organization (*gatekeeper.dec.com, whitehouse.gov*)**

What are the threats?

- Curious crackers
  - Just poking around to see what they can get into
- Vandals
  - System downtime
  - Network outages
  - Telephone line use
- Industrial spies
  - Loss of trade secrets
  - Loss of competitive information
  - Loss of reputation
- Accidental data disclosure
  - Employee privacy rights
  - Client privacy rights

Purpose of the Toolkit

- Provide a freely available set of tools for building internet firewalls
- Focus on components that are flexible enough to be useful in different types of firewall configurations
- Improve security for the Internet community by providing a common starting point that is widely used and understood
- Provide a centerpiece for further development and enhancement

What is the Toolkit?

- Design Philosophy
  - A framework for thinking about firewalls
  - Based on practical experience
- Configuration Practice / Verification Strategy
  - Approaches to systems management that minimize risk
  - Software and system security that is testable
- Software Tools
  - Proxy servers for FTP, TELNET, rlogin, NNTP, X-Window, HTTP(WWW), Gopher
  - Compartmented SMTP exchanger
  - User authentication server
  - TCP service access control
  - Common configuration for all components

Design Philosophy

- *KISS*
- Complex applications more likely to have bugs and bugs are harder to find
- Run services without permissions if possible

Combining complex applications with privileged processing invites disaster

- Isolate processes from the system

On the firewall, prevent untrusted systems from ever connecting to privileged processes that have not been chrooted

Network Security Paradigms

- That which is not expressly permitted is prohibited
  - Firewall blocks everything
  - Administrator must take steps to support each service
  - Implicitly “dis-empowers” users

- That which is not expressly prohibited is permitted
  - Firewall blocks services that are known potential security risks
  - Users can potentially introduce security holes in system

Perimeter Defense

- Principles of Perimeter Defense:
  - Protect all access paths leading into the network, using internet firewalls, passwords on terminal servers, modem callback, etc.
  - Within the network, hosts can trust each other
  - Not concerned about ethernet sniffer attacks within the network
  - "Crunchy shell around soft, chewy center" *(Bill Cheswick, AT&T)*

Defense In Depth

- Principles of Defense in Depth:
  – Protect all access to hosts, enforcing strict security policies on each system that is on network
  – Within the network, hosts require strong authentication for trust (Kerberos, SecurID, etc)
  – Restrict connection to the network - users generally do not manage or install their own systems

A Threat Model for Firewalls

- Threat is quantified in terms of “Zones of Risk”
  - How many potential angles of attack can the firewall be subjected to?
  - What is the progression of attacks if the first one succeeds?
  - At what point do an attacker’s actions become untraceable?
  - At what point do the network’s defenses completely come apart?

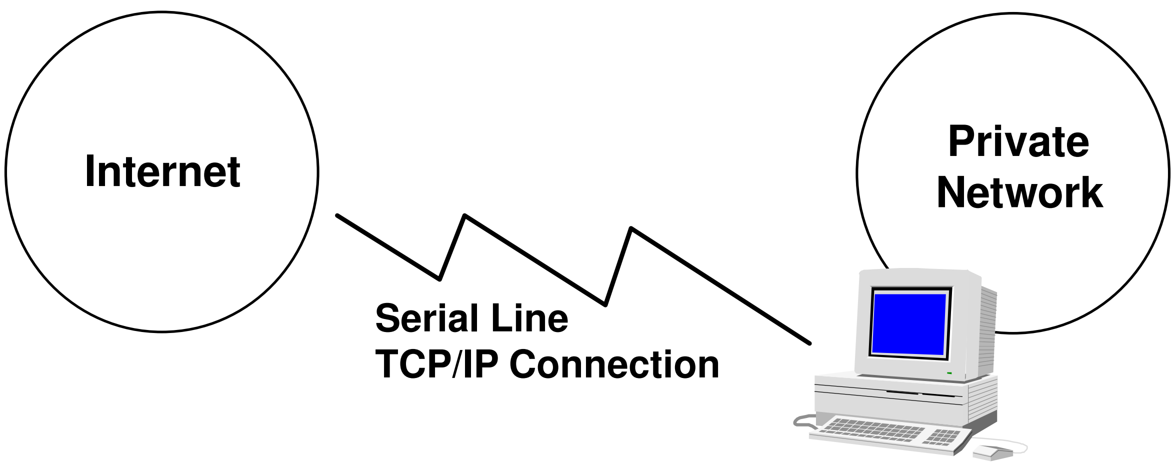
A Threat Model for Firewalls

- Minimizing zones of risk entails restricting the "size" of the network connection to be as small as possible
- Single point of security failure must be avoided if possible

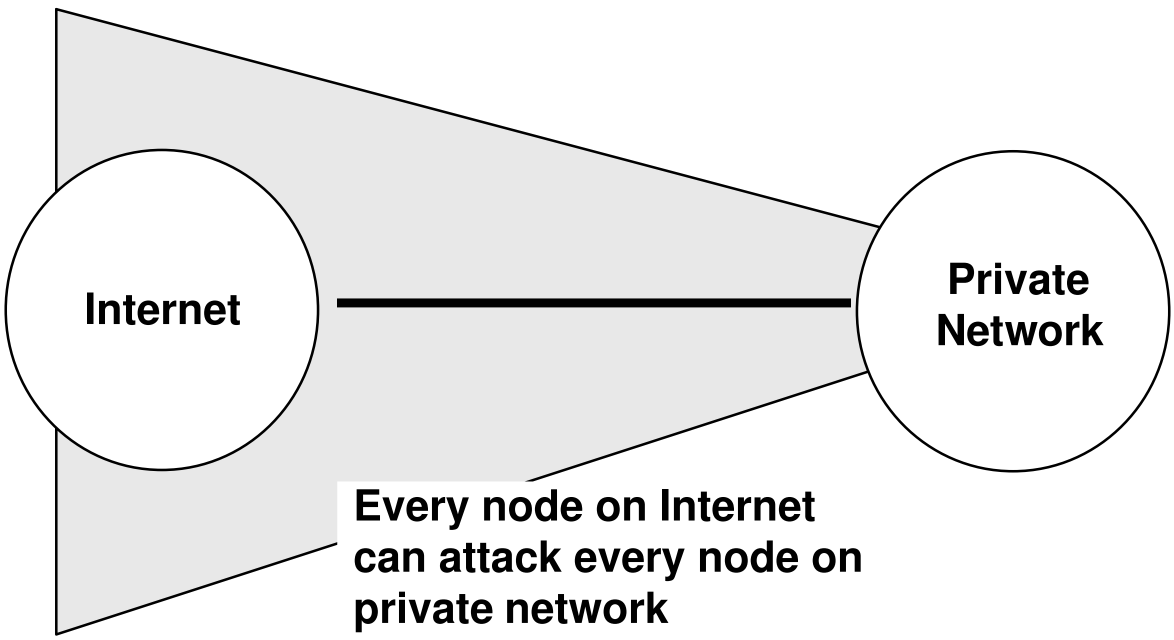
No Internet Connection

No Internet Connection

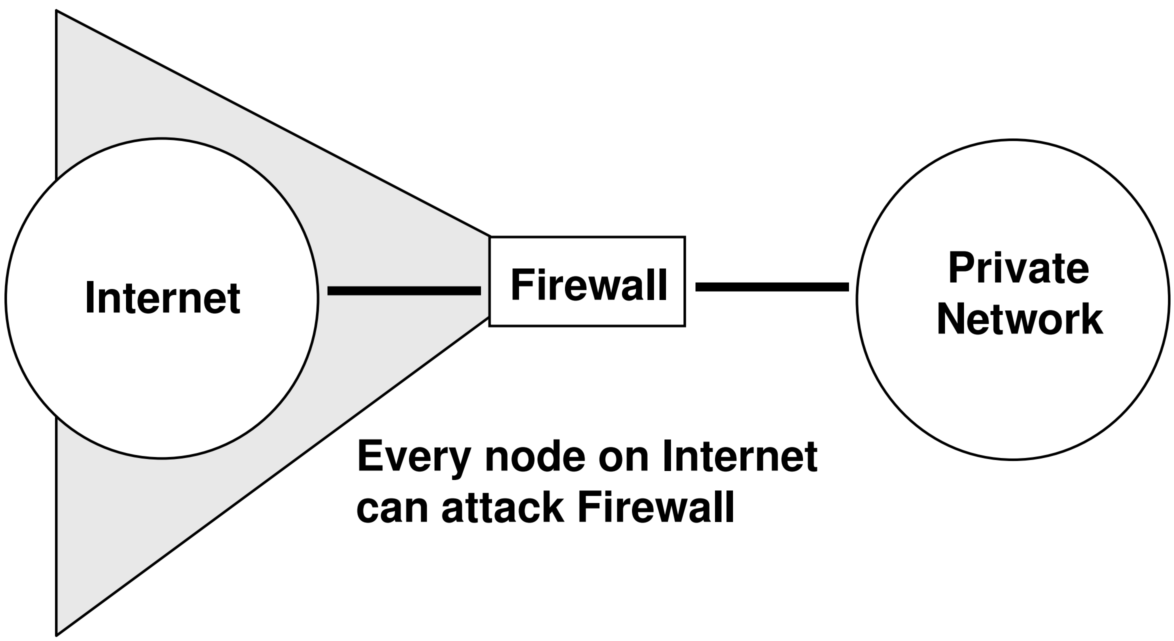
....And a single enterprising user



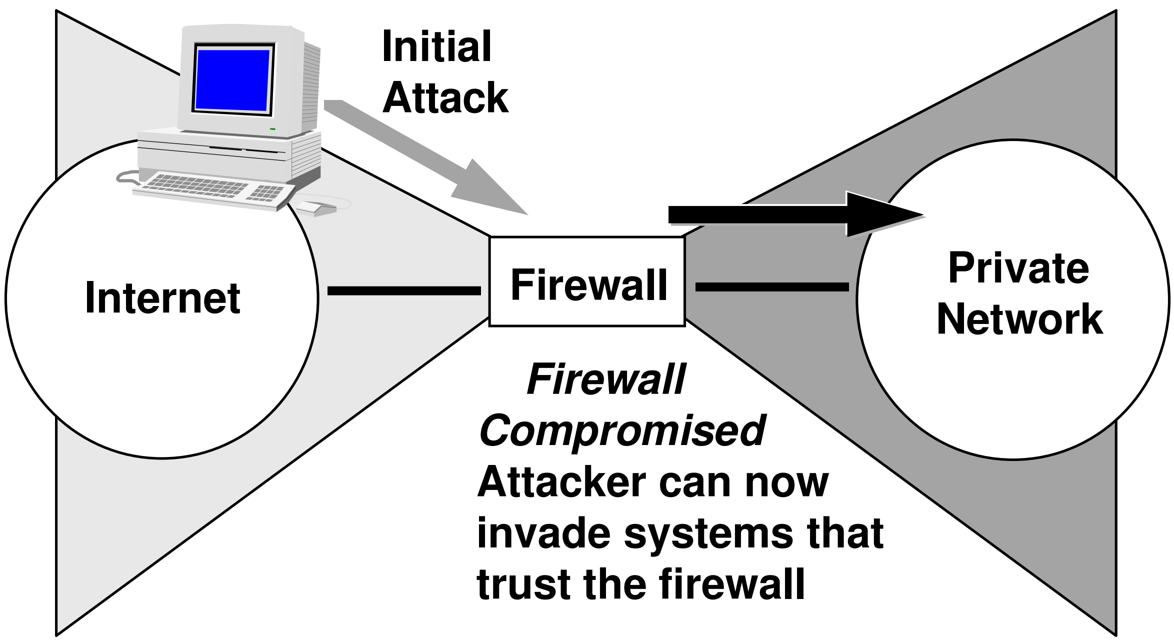
Direct Internet Connection



Zone of Risk With a Firewall



Island-Hopping Attacks



The User Problem

- **Most systems break-ins are result of users weakening security**
- **Most system security bugs require a log-in on the system to exploit them fully**

Solution: ***Keep users off the firewall itself!!***

- **This means that someone logging into the firewall is a noteworthy security event**
- **Reduces systems management requirements**

Screening Routers

- Can be a commercial router (Cisco, Wellfleet, Proteon, etc)
- Can be a host running an operating system that supports packet screening
- Some screening routers permit various levels and types of packet logging
- Many firewalls consist of nothing more than a screening router

“Established” Connections

- Many routers permit “established” connections
- TCP traffic with SYN bit set in packets is indicative of established connection
- Permit only traffic with SYN bit set to <u>enter</u> network
- Connections from “inside” can originate to “outside” systems and return traffic permitted because SYN bit is set
- FTP still will not work properly
  - Many sites modify FTP client to bind specific range of ports and then permit incoming traffic on those ports

Risks of Screening Routers

- Very minimal logging information
  - Virtually no audit trail other than traffic statistics
- Hard to get screening rules
- Firmware bugs can be undetectable and leave entire network open to attack
- Temptation to open “holes” in screen for new services increases likelihood of error
- Services can be “tunnelled” on top of other services to bypass the firewall (implementing remote terminal access via name server traffic)
- Impossible to add authentication

Source-routed attacks

- IP options permit traffic to specify route a packet should take in the header
- Unless router is configured to screen traffic with IP options set someone can generate traffic from “outside” that appears to have originated from “inside”
- If systems establish trust based on network address administrators should ensure that source routing is blocked
- Some older versions of router firmware did not implement this screening properly and many networks were vulnerable

DNS Attacks

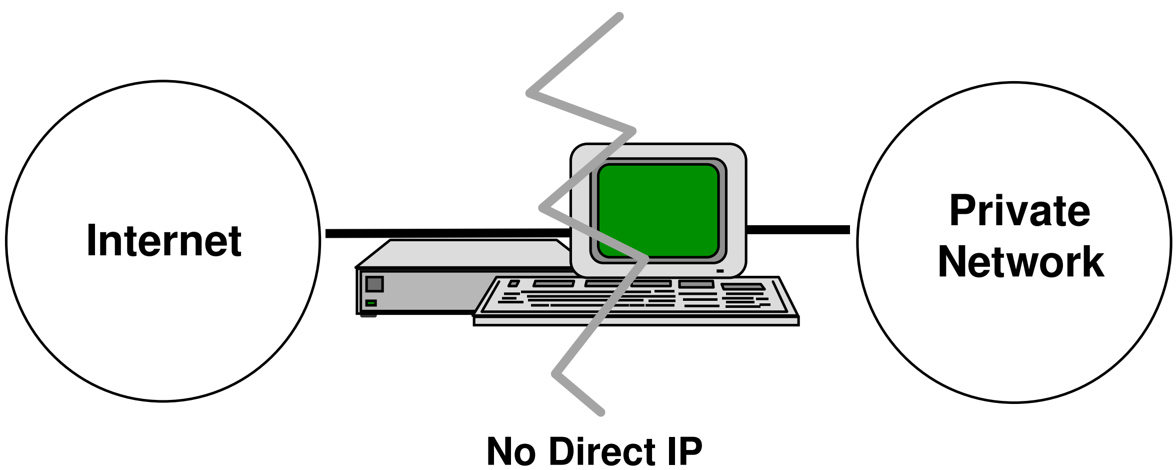
- Many applications rely on host names for security (rlogin, NFS)
- DNS (Domain Name System) maps network addresses to host names and vice versa
- Systems running DNS that rely on host names for security may be fooled by attackers who provide bogus entries to name servers
  - Name servers cache entries
  - Very little security on name server traffic (a simple query ID)
- Once system has spoofed address entry it may masquerade as a trusted host

Bastion Hosts

- **Bastion Hosts are systems identified as network "strong points"**
- **Often act in capacity of Email relays, name servers, FTP servers, USENET News servers, etc.**
- **Generally, a Bastion Host is one that is recognized as a potential point of attack (*whitehouse.gov* or *dockmaster.ncsc.mil*)**

Dual Homed Gateways

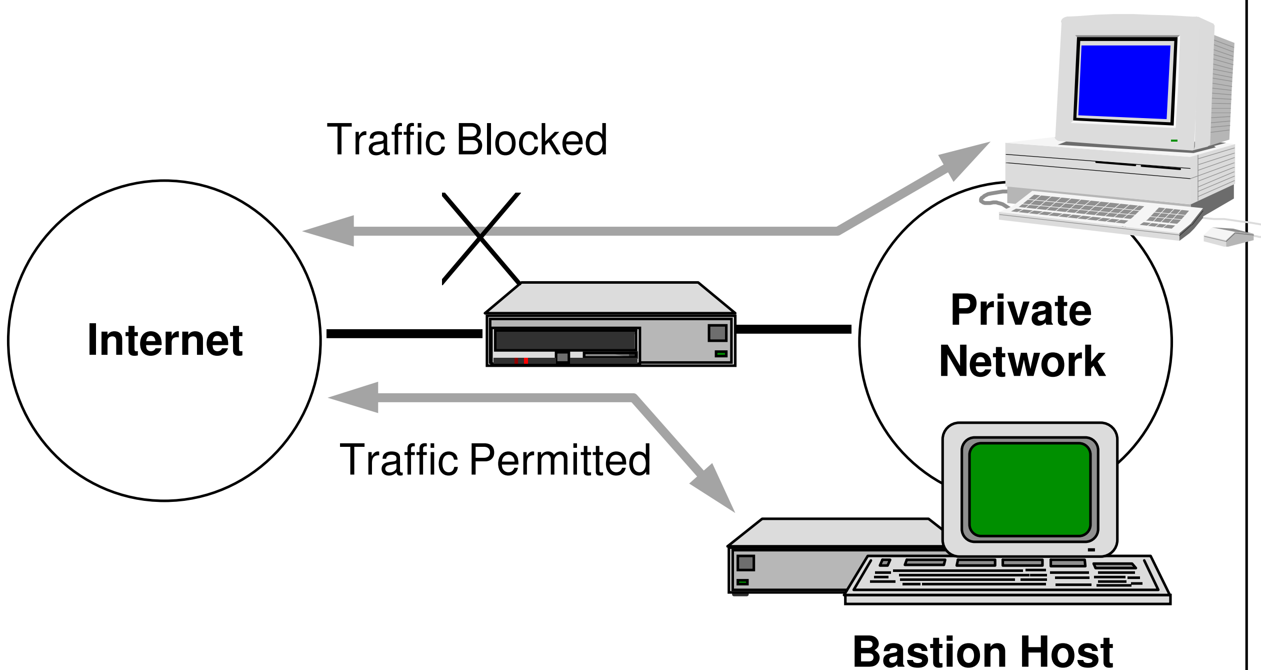
- Dual Homed Gateways are based on a system with two network connections, with routing between them disabled



Dual Homed Gateways

- Reachable from both Internet and private network, but traffic cannot flow across it directly
- Some sites use a Dual Homed Gateway as a user “base of operations” for access to Internet
  - Users must first log in to Gateway, then can log in to hosts on Internet, or do FTP, etc.
- Dual Homed Gateway system is, by definition, a Bastion Host

Screened Host Gateways

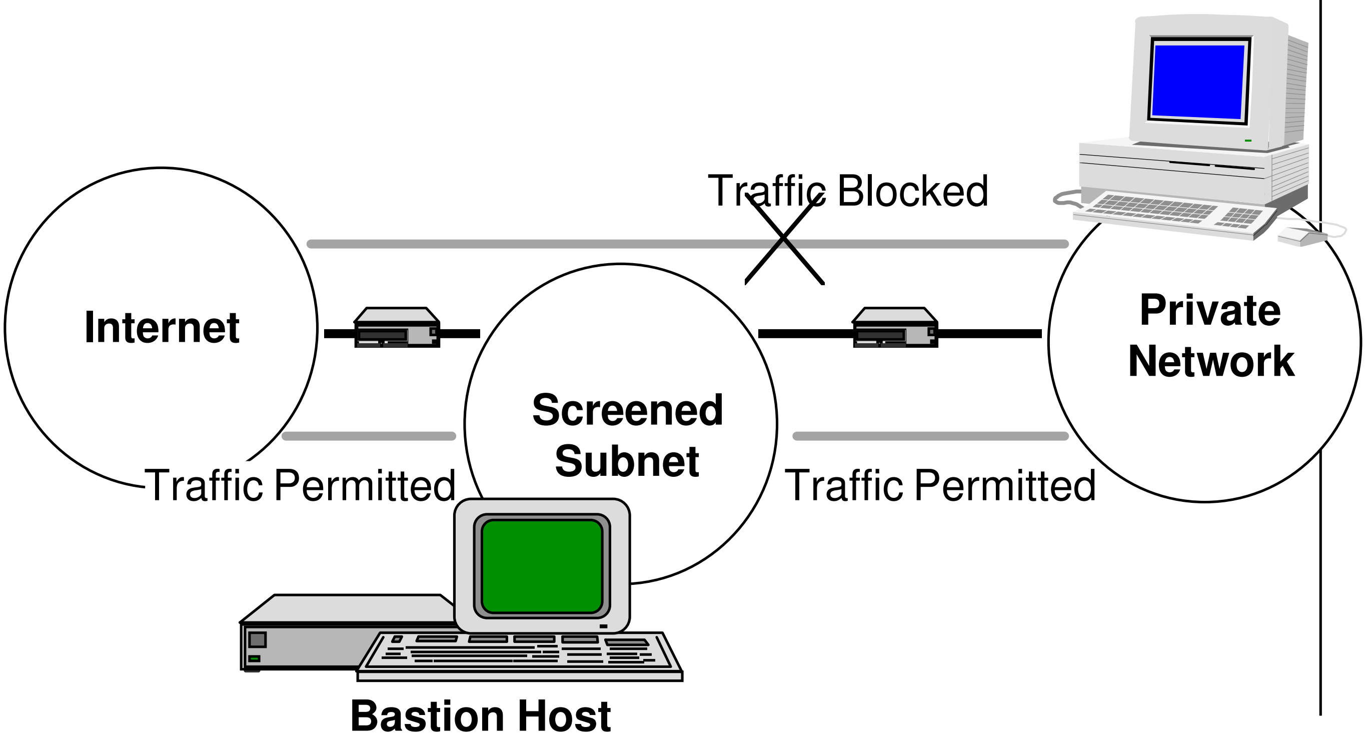


Screened Host Gateways

- **Screened Host Gateways are the most common form of firewall, also the most flexible**
- **Screening Router blocks traffic between Internet and all hosts on private network *except* for a single Bastion Host**
- **Screening Router can be configured to permit nodes on private network to directly access Internet via TELNET or FTP**

tis

Screened Subnets

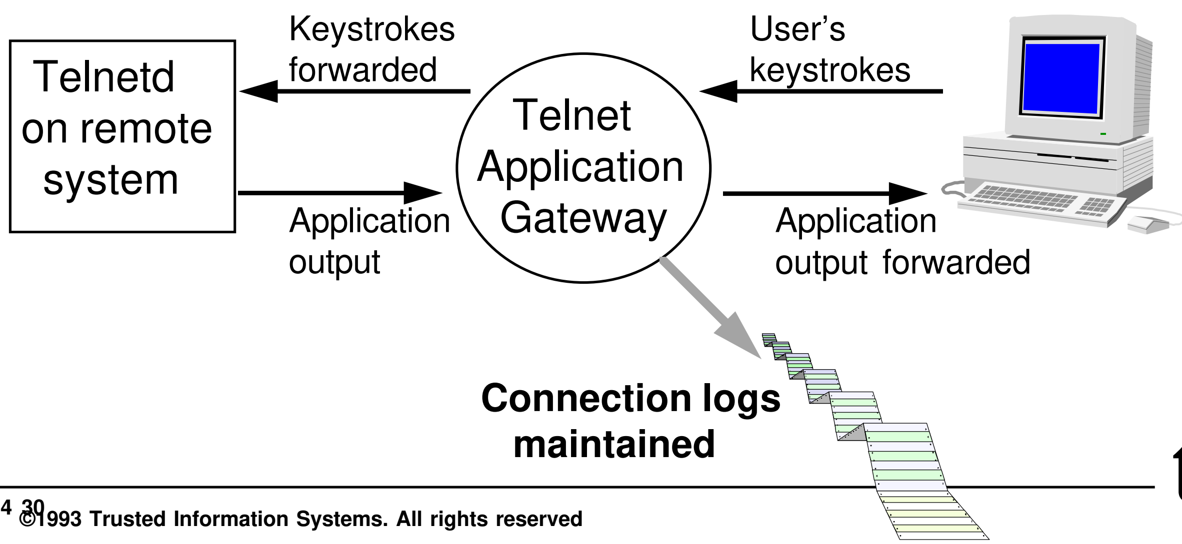


Screened Subnets

- Screened Subnet approach places a "sandbox" network between the Internet and the private network
- Internet can only communicate with nodes on the Screened Subnet
- Private network nodes can only communicate with nodes on the Screened Subnet
- Permits the private network to be effectively "invisible" to the Internet

Proxies / Application Gateways

• Application Gateways handle store and forward traffic and some types of interactive traffic



Proxies / Application Gateways

- Since Application Gateways handle traffic at an application level, they understand the protocol of that application, and can log/audit traffic
- Application Gateways can have extra security or authorization built into them as needed
- Examples:
  - Sendmail (arguably)
  - Telnet gateway (tn-gw)
  - FTP gateway (ftp-gw)
  - X11 protocol forwarder (x-gw)
  - HTTP/Gopher proxy (http-gw)

Toolkit Proxies

- **Tn-gw**
  - TELNET protocol proxy
  - May be configured to require authentication based on origin of connection
  - All connections and amount of data transferred are logged
- **Rlogin-gw**
  - Rlogin proxy
  - May be configured to require authentication based on origin of connection
  - Supports transparent pass-through with *user@host*
  - All connections and amount of data transferred are logged

tis

Toolkit Proxies *(cont)*

- **Ftp-gw**
  - FTP protocol proxy
  - Can require authentication based on origin of connection and type of operation requested
  - May selectively block FTP commands based on origin of connection
  - All connections and traffic logged
  - May selectively log and summarize specified FTP commands
- **Plug-gw**
  - Generic TCP “plug-board” proxy
  - Useful for transparently gatewaying USENET news through a firewall

Toolkit Proxies *(cont)*

- **X-gw**
  - X-Window protocol proxy
  - Invoked from TELNET or Rlogin proxy without giving user access to firewall system
  - May be selectively enabled or disabled depending on destination
  - May require authentication
  - Causes a “pop-up” window to appear on user’s workstation requesting permission before permitting new X clients to connect to the display server
  - May be used to provide X access from inside to outside, or from outside to inside

Toolkit Proxies *(cont)*

- **Http-gw**
  - Combined HTTP (WWW/Mosaic) proxy and Gopher proxy
  - Recognizes and supports Mosaic clients with proxy forwarder extensions
  - Also supports Mosaic clients without proxy extensions by re-writing URLs to route requests through the proxy
  - Minimalist implementation
    - » Under 3,000 lines of source code (as opposed to approximately 60,000 for other HTTP forwarders)
  - Future support for authentication and link level encryption

Tn-Gw Operation Example

```
%->
%-> telnet gatekeeper
Trying 192.33.112.117 ...
Connected to gatekeeper.
Escape character is '^]'.
gatekeeper telnet proxy (Version V1.0) ready:
tn-gw-> help
Valid commands are: (unique abbreviations may be used)
   connect hostname [port]
   help/?
   quit/exit
tn-gw-> c somebox.domain


SomeOS UNIX (somebox)

login: you
Password: ######
Last login: Mon Sep 27 21:22:16 from some.other.box
You have 5 new mail messages
somebox% logout
%->
```

Rlogin-Gw Operation Example

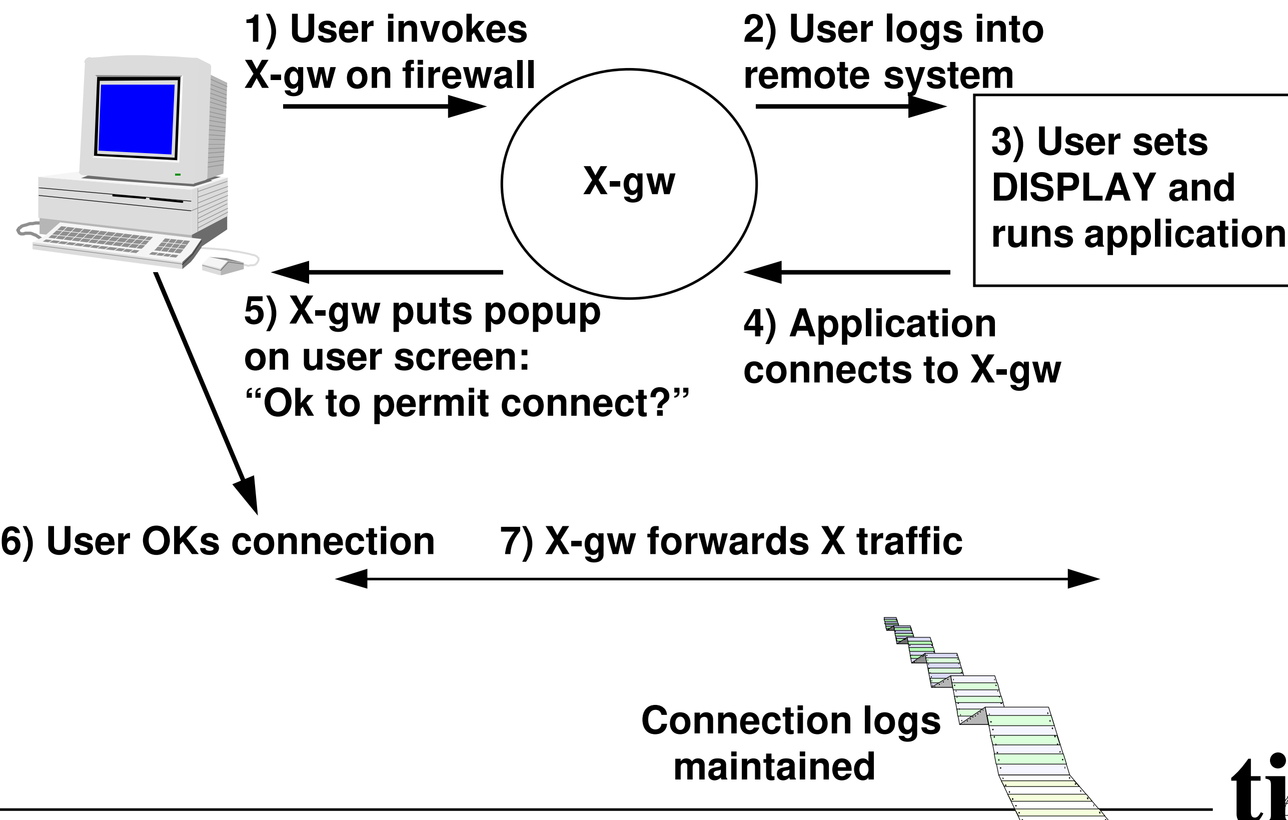
```
%->
%-> rlogin gatekeeper -l you@somebox.domain
Username: you
Password: ######
Login Accepted
(Connected to somebox.domain via proxy)
Last login: Mon Sep 27 21:22:16 from some.other.box
You have 5 new mail messages

somebox-> logout
Connection closed.
%->
%-> rlogin gatekeeper
Username: you
Password: ######
Login Accepted
rlogin-gw-> ?
Valid commands are: (unique abbreviations may be used)
   connect hostname
   help/?
   quit/exit
rlogin-gw-> quit
Connection closed.
%->
```

FTP-Gw Operation Example

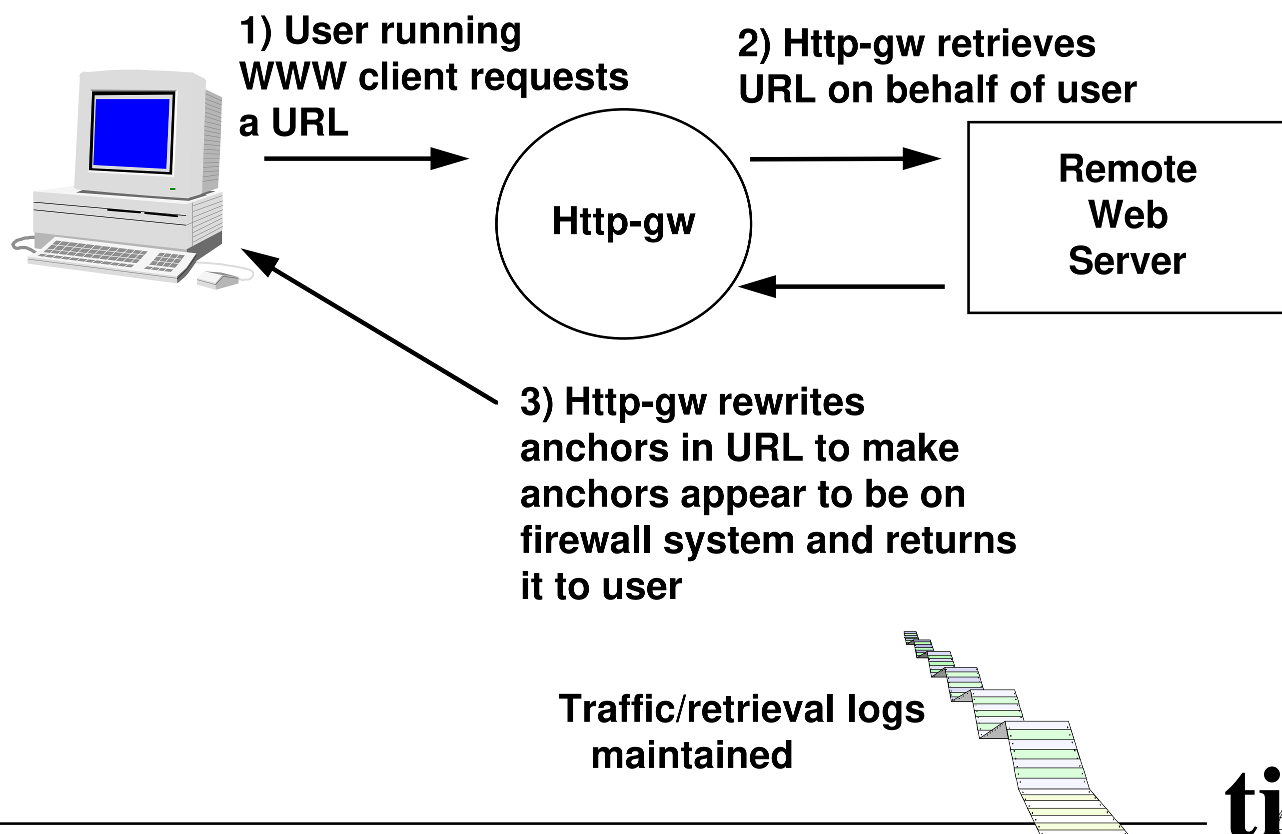
```
%->
%-> ftp gatekeeper 1555
Connected to gatekeeper.
220 gatekeeper FTP proxy (Version 1.0 stable) ready.
Name (gatekeeper:you): you@somebox
331-(----GATEWAY CONNECTED TO somebox----)
331-(220 somebox FTP server (Version 5.6/mjr) ready.)
331 Password required for you.
Password: ######
230 User you logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get myfile
200 PORT command successful.
150 Opening BINARY connection for myfile(2048 bytes).
226 Transfer complete.
2048 bytes received in 1e-06 seconds (2e+06 Kbytes/s)
ftp> quit
221 Goodbye.
%->
%->
```


tis

X-Gw Operation Example



Http-Gw Operation Example



Logging

- It's easy to log too much
- Log interesting transactions
- Use simple ad-hoc data reduction tools to automatically search for interesting events to bring to systems manager's attention
  - More detailed searches can be performed if a problem or interesting condition is identified
- Disk space is cheap
  - Several months worth of logs only uses a few hundred megabytes of disk compressed

It's easier to throw logging information away after you've decided it's useless than it is to get it back, if you didn't save it in the first place

Sample Log Output

```
ftp-gw[3442]: permit host=sol/192.33.112.100 use of
    gateway
ftp-gw[3442]: exit host=sol/192.33.112.100 cmds=0 in=0
    out=0 duration=2
ftp-gw[3875]: permit host=sol/192.33.112.100 use of
    gateway
ftp-gw[3875]: sol/192.33.112.100: RETR myfile
ftp-gw[3875]: sol/192.33.112.100: STOR outfile
ftp-gw[3875]: exit host=sol/192.33.112.100 cmds=9
    in=412311 out=93191
```

```
rlogin-gw[3877]: permit host=sol/192.33.112.100 use of
    gateway
rlogin-gw[3877]: exit host=sol/192.33.112.100 no auth
rlogin-gw[3945]: permit host=otter/192.33.112.117 use of
    gateway
rlogin-gw[3945]: authenticate user=you
rlogin-gw[3945]: connected host=otter/192.33.112.117
    to=homebox
rlogin-gw[3945]: exit host=otter/192.33.112.117 dest=sol
    in=21224 out=1320
```

Strong User Authentication

- Passwords should never be transmitted in clear over untrusted networks
- Ideal security combines something secret that a user ***knows*** with something physical that a user ***has*** in their possession
- Requires applications to support chosen authentication mechanism
  - Can be costly to deploy organization-wide
  - Firewall is a natural “choke point” for requiring authentication on a per-service basis
- Often hard to integrate with existing hardware base (terminal servers)
- Lack of standards

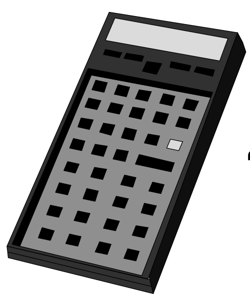
Authentication Tokens

Server sends
Challenge: "876261"

User "unlocks" authentication token by keying in Personal Identification Number

"1105"

(Server and token share a secret encryption key that user never has access to)



"876261"

User enters challenge into authentication token

(Secret key: "0x8A5F42")

"722512"

Token returns an encrypted response to the challenge

Server compares response

"722512"

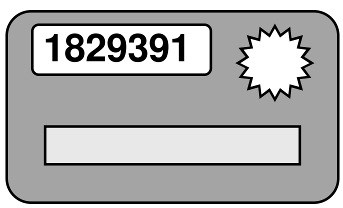
User responds to server with unique response

Authentication Tokens

(Server and token share a secret encryption key that user never has access to)

“1105”

User “unlocks” authentication token by keying in Personal Identification Number



(Secret key: “0x1A904C”)
Card also has internal clock

Token displays a number based on its internal clock, the user’s PIN, and the secret encryption key

Server compares response based on system clock, PIN, and secret key

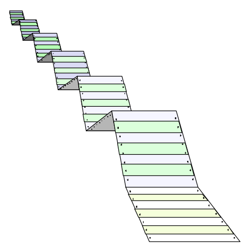
“1829391”

User responds to server with encrypted response

tis

Authentication Techniques

Server and user share a secret encryption key



User generates hard copy list of challenges and responses based on cryptographic hash of secret key

Server challenges user with a list entry:
“Tell me entry #34”

User consults hard copy list and locates entry #34

Entry #34 is never used again

User responds to server with matching response string

“otter chomp eel marble”

Server compares response from its database

The Authentication Server

- Many good forms of authentication available
  - All have different programming interfaces
  - Some do not work over networks
  - Some license per node
- For large organizations it can be too costly to give everyone an authentication token
- Sometimes separate administrative domains want to use the same firewall

Solution: ***The authentication server acts as "middleware" and manages multiple forms of authentication***

The Authentication Server

- Each user has a record containing:
  - Last login time
  - Number of bad logins
  - User's group membership
  - Type of authentication to perform with user
- Selected users (group administrators) may be given "ownership" of a group
  - May add or delete users from their group
  - May enable or disable members of their group
  - May list group members
- Database administrator
  - May create groups and group administrators
  - Has complete control over all user records

Proxy Authentication Example

```
untrusted.site->
untrusted.site-> rlogin gatekeeper.home.com
Username: you
Challenge "624072": 813212
rlogin-gw-> c homebox
(Connected to homebox via proxy)
Last login: Mon Sep 27 21:22:16 from some.other.box
You have 5 new mail messages
homebox% logout
Connection closed.
untrusted.site->
untrusted.site->
```

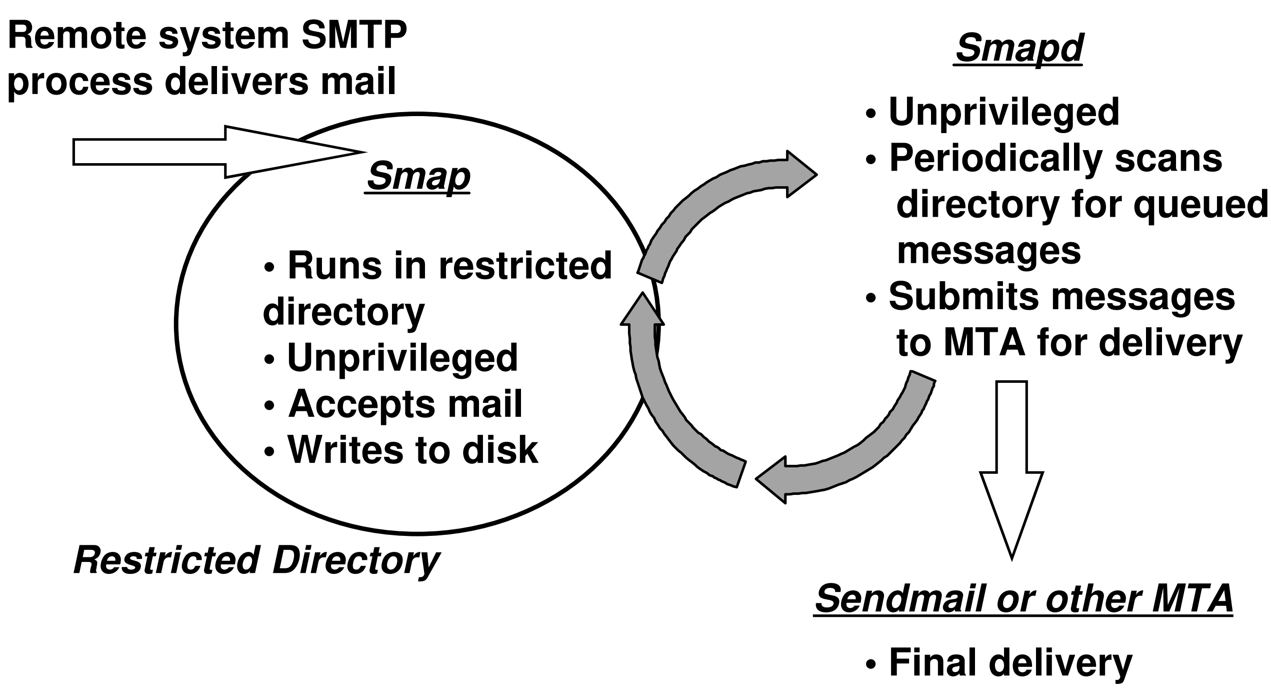
Compartmenting SMTP

- **Many mail transfer agents are large complex applications that run with permissions**
- **Often implicated in system break-ins (Morris Internet worm)**

***Solution:* Isolate mail transfer to a chrooted unprivileged process**

- **Toolkit includes *smap*, an SMTP listener**
  - **Runs chrooted in an isolated directory as an unprivileged user**
  - **Gathers messages and saves them to disk**
  - **Simple implementation is easy to verify for correctness**
- ***Smapd* daemon process**
  - **Sweeps directory and submits mail to MTA for delivery**

Smap / Smapd


Remote system SMTP process delivers mail
Smap
• Runs in restricted directory
• Unprivileged
• Accepts mail
• Writes to disk
Restricted Directory
Smapd
• Unprivileged
• Periodically scans directory for queued messages
• Submits messages to MTA for delivery
Sendmail or other MTA
• Final delivery

TCP Service Access Control

- On UNIX systems, *inetd* spawns processes to service connections upon receipt of request
- "TCP wrappers" are invoked instead of normal service process
  - Wrapper program checks origin address
  - Wrapper may log connection
  - Wrapper program either invokes the proper service process or exits, depending on whether the origin address is permitted
- Firewall toolkit includes *netacl*
  - Generic wrapper program
  - Minimalist implementation
  - Shares common configuration file with rest of toolkit

Netacl and Inetd

New service request

Inetd

- Listens for and accepts connections
- Invokes service process

netacl

?

- Determines caller's network address
- Checks permissions
- Logs connection

Invoke
Server process

Disconnect

Configuration and Control

- All firewall components share common configuration file
- Components use configuration file rather than compile-time options or command line parameters
  - Easy to manage
  - Easy to change
  - Easy to read
  - One syntax to learn
- When applications are started, configuration information is read and stored in memory
  - Configuration options still work after chroot

Configuration File

```
#netacl configuration rules
netacl-in.telnetd: permit-hosts 192.33.112.* -exec \
    /usr/etc/in.telnetd
netacl-in.rshd: permit-hosts 192.33.112.* -exec \
    /usr/etc/in.rshd
netacl-in.ftpd: permit-hosts 192.33.112.* -exec \
    /usr/etc/in.ftpd
netacl-in.ftpd: permit-hosts unknown -exec /bin/cat \
    /usr/local/etc/noftp.txt
netacl-in.ftpd: permit-hosts * -chroot /home/ftp -exec \
    /bin/ftpd -f -l
netacl-in.fingerd: permit-hosts 192.33.112.* -exec \
    /usr/etc/in.fingerd
netacl-in.fingerd: permit-hosts * -exec /bin/cat \
    /usr/local/etc/finger.txt

# smap configuration rules:
smap, smapd:    userid 4
smap, smapd:    directory /mail/inspool
smapd:          executable /usr/local/etc/smapd
smap:           logfile log
smap:           maxrecip 4000
smap:           maxbytes 1048576
smap:           timeout 3600
```

Toolkit Documentation

- **Overview guide**
  - High level description of toolkit and design principles
  - Intended for general audience
- **Administrator's guide**
  - Technical discussion of configuration and installation issues and procedures
  - Intended to help explain installation
- **User's guide**
  - Brief handout explaining how to use proxies
  - Intended to help users familiarize quickly with toolkit
- **Manual pages**
  - Reference documentation for components and configuration options

Buying Versus Building

- **Vendor Packaged Firewalls**
  - + Good Support and documentation
  - - May be expensive
- **Consultant Firewalls**
  - + Tailored to meet your needs
  - - Support may be issue
  - + Costs vary based on functionality
- **Home Brew Firewalls**
  - + Sometimes cost effective
  - - Requires in-house expert (what if the expert leaves?)
  - + Tailored to meet your needs
  - - Possible security holes may be overlooked

A “Crystal Box” Solution

- Black Box
  - Vendor installs it and configures it
  - Software is proprietary
  - Sometimes requires vendor-specific hardware
  - “Trust Us” security model
  - Good support and maintenance
- Crystal Box
  - Install it yourself or have a consultant install it
  - Source code and documentation available free
  - Runs on many platforms
  - Software can be examined for security flaws

tis

Conclusions and Observations

- **Don't be scared of the Internet**
- **Build on past experience - there are many examples of good firewalls out there**
  - **Firewalls mailing list on Internet (*firewalls@greatcircle.com*, send mail to *majordomo@greatcircle.com* to get on it)**
  - **Papers by AT&T Research on *research.att.com* in *dist/internet_security***
  - **Papers on *tis.com* in *pub/firewalls***
- **Commercial and consulting offerings for firewalls are plentiful**

Toolkit Availability

- **Available over the Internet in source code form**
  - **FTP from *ftp.tis.com   pub/firewalls/toolkit/fwtk.tar.Z***
- **“Open” solution**
  - **Independent consultants may use it for customer solutions**
  - **Components may be licensed to be repackaged and sold in binary form**
  - **Contributions of software and documentation accepted from users**
- **Ongoing evolution**
  - **New components will be added over time**
  - **Ongoing bug-fixes and refinement**

Sure, I'm paranoid.

But am I paranoid *enough*?