

ECE-GY 9383

# Special Topics in Network Security

## 8 - Network Defense

---

Slides credits: Shivendra S. Panwar, Fraida Fund

CSE/ECE zjzhao



**NYU**

**TANDON SCHOOL  
OF ENGINEERING**

# In this lecture

---

- Firewalls
- Intrusion detection
- Examples of intrusion prevention/defense system

# Network defense

- Keep attacker out (perimeter defense)
- Detect when attacker gets in (intrusion detection)
- Respond to attack (not covered in this lecture)

[Reference: Stallings, Chapter 11, 12]

# Firewalls

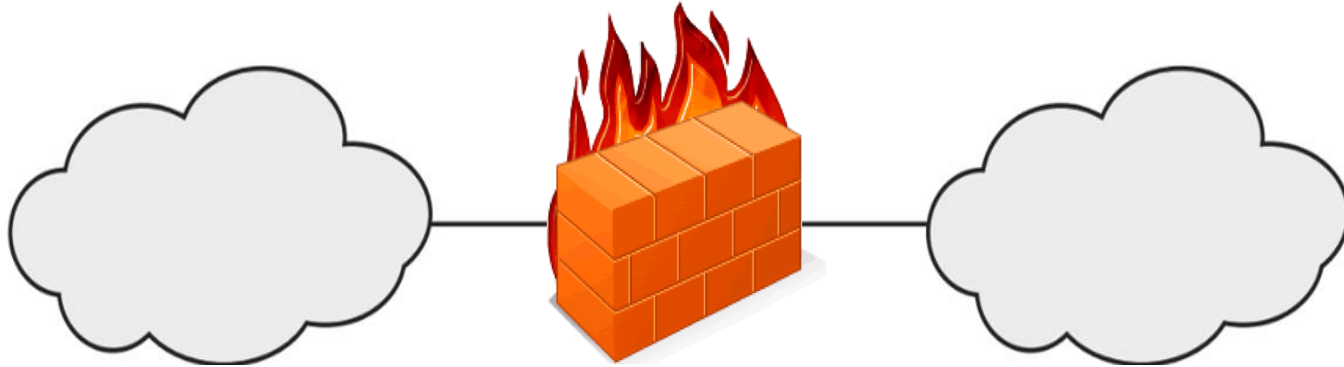
---

# General firewall model

External (untrusted) network  
(e.g. Internet)

Firewall

Internal (protected) network  
(e.g. enterprise network)

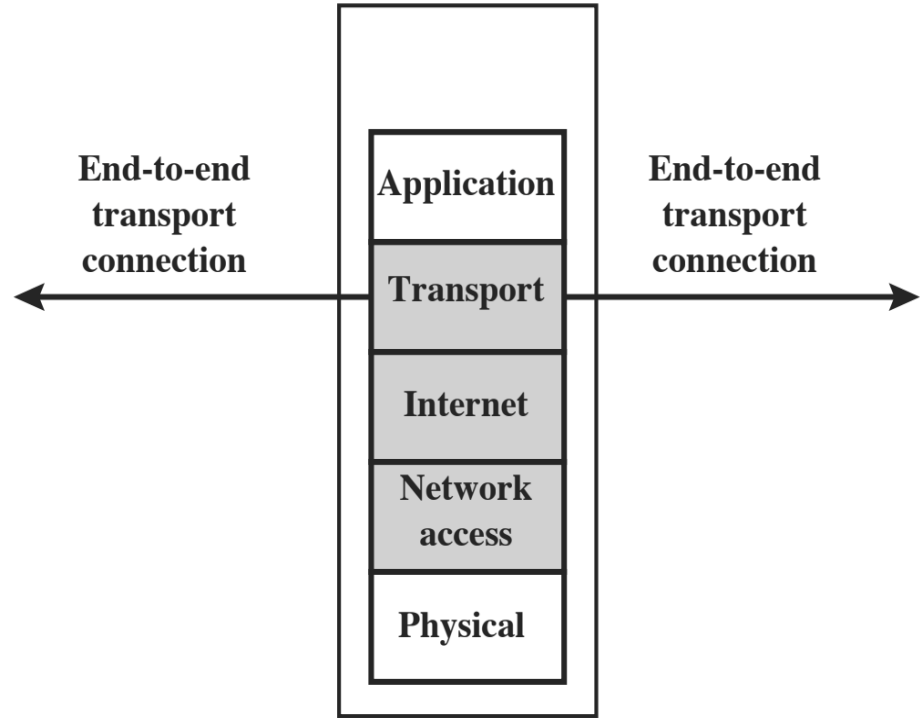


- All traffic between external network and internal networks must pass through the firewall
- Only authorized traffic, as defined by the local security policy, will be allowed to pass
- The firewall itself is immune to penetration

# Firewall characteristics

Filter traffic by

- IP address and protocol (network and transport layer headers)
- Application protocol data (e.g. check SMTP email for spam)
- User identity
- Network activity (time of day, rate of request)



**(b) Packet filtering firewall**


# Packet filtering example (table)

**Table 12.1 Packet-Filtering Example**

<b>Rule</b>	<b>Direction</b>	<b>Src address</b>	<b>Dest addresss</b>	<b>Protocol</b>	<b>Dest port</b>	<b>Action</b>
A	In	External	Internal	TCP	25	Permit
B	Out	Internal	External	TCP	> 49151	Permit
C	Out	Internal	External	TCP	25	Permit
D	In	External	Internal	TCP	> 49151	Permit
E	Either	Any	Any	Any	Any	Deny

# Packet Filtering firewalls

## Weaknesses

- 
- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions
  - Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited
  - Most packet filter firewalls do not support advanced user authentication schemes
  - Packet filter firewalls are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack
  - Due to the small number of variables used in access control decisions, packet filter firewalls are susceptible to security breaches caused by improper configurations



## Strengths

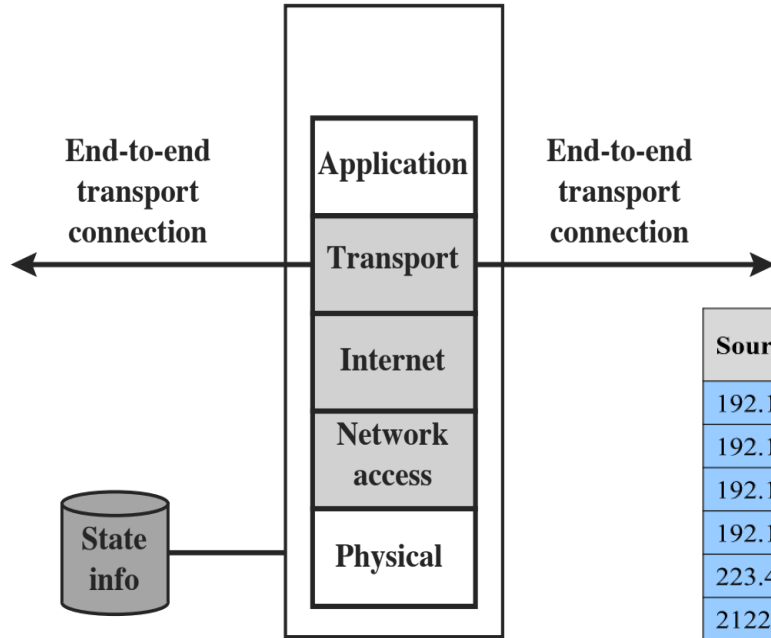
- Its simplicity
- Transparent to users and are very fast



# Attacks on packet filtering firewall

- IP address spoofing
- Source routing attacks
- Tiny Fragment and Fragment Overlap attacks (split TCP header)
  - Example: firewall blocks TCP port 23 but allows port 25
  - Send first packet with MF (More Fragments) bit set and destination port 25
  - Send second packet with fragmentation offset = 1 (overwrites all but first eight bytes of the first packet), and destination port 23
  - Firewall ignores second packet's TCP header and allows all fragments through; host reassembles fragments and delivers to port 23

# Stateful firewall



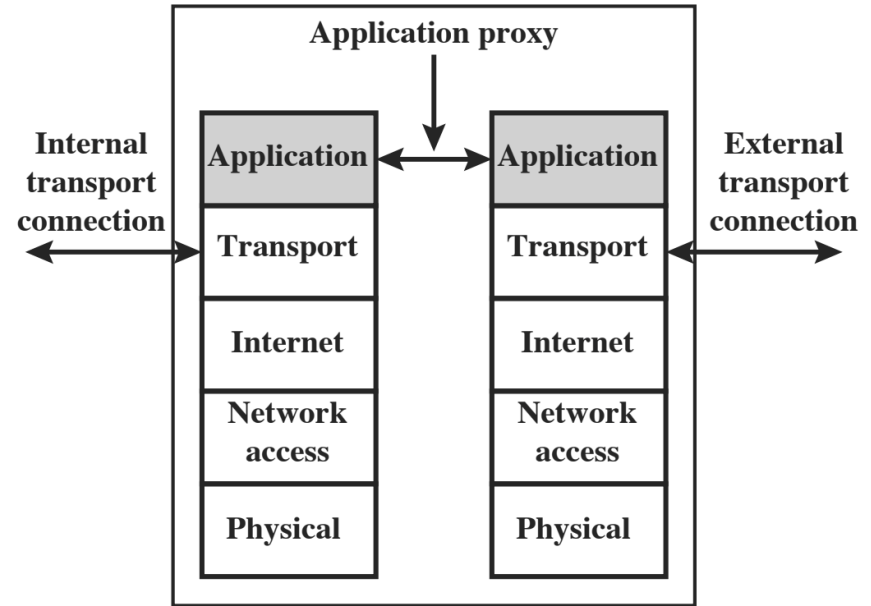
(c) Stateful inspection firewall

Table 12.2 Example Stateful Firewall Connection State Table [SCAR09b]

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	51030	210.22.88.29	80	Established
192.168.1.102	51031	216.32.42.123	80	Established
192.168.1.101	51033	173.66.32.122	25	Established
192.168.1.106	51035	177.231.32.12	79	Established
223.43.21.231	51990	192.168.1.6	80	Established
2122.22.123.32	52112	192.168.1.6	80	Established
210.922.212.18	53321	192.168.1.6	80	Established
24.102.32.23	51025	192.168.1.6	80	Established
223.21.22.12	51046	192.168.1.6	80	Established

# Application Level Gateway

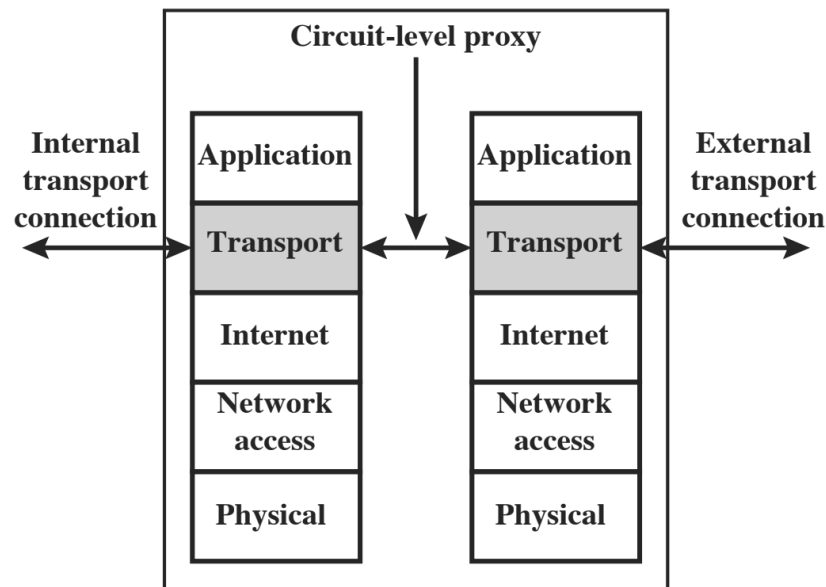
- Also called an application proxy
- Acts as a relay of application-level traffic
- If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall
- The gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features
- Tend to be more secure than packet filters
- Disadvantage: The additional processing overhead on each connection



**(d) Application proxy firewall**

# Circuit-Level Gateway

- Also called *circuit-level proxy*
- Can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications
- Does not permit an end-to-end TCP connection
- The security function consists of determining which connections will be allowed
- Typical use is a situation in which the system administrator trusts the internal users
- Can be configured to support application-level or proxy service on inbound connections and circuit-level functions for outbound connections



(e) Circuit-level proxy firewall

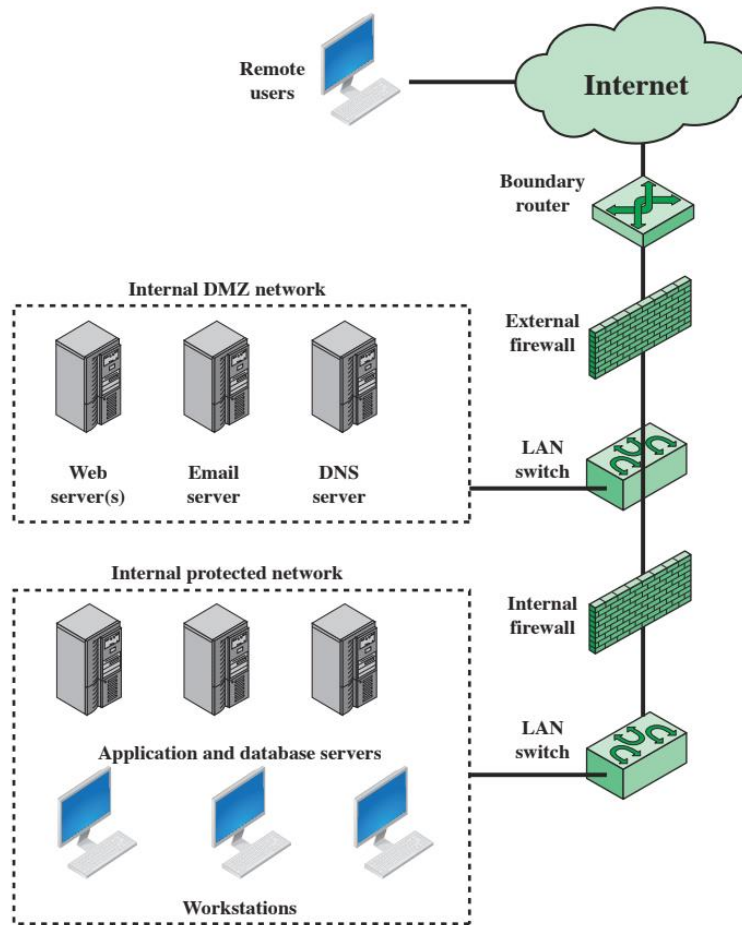
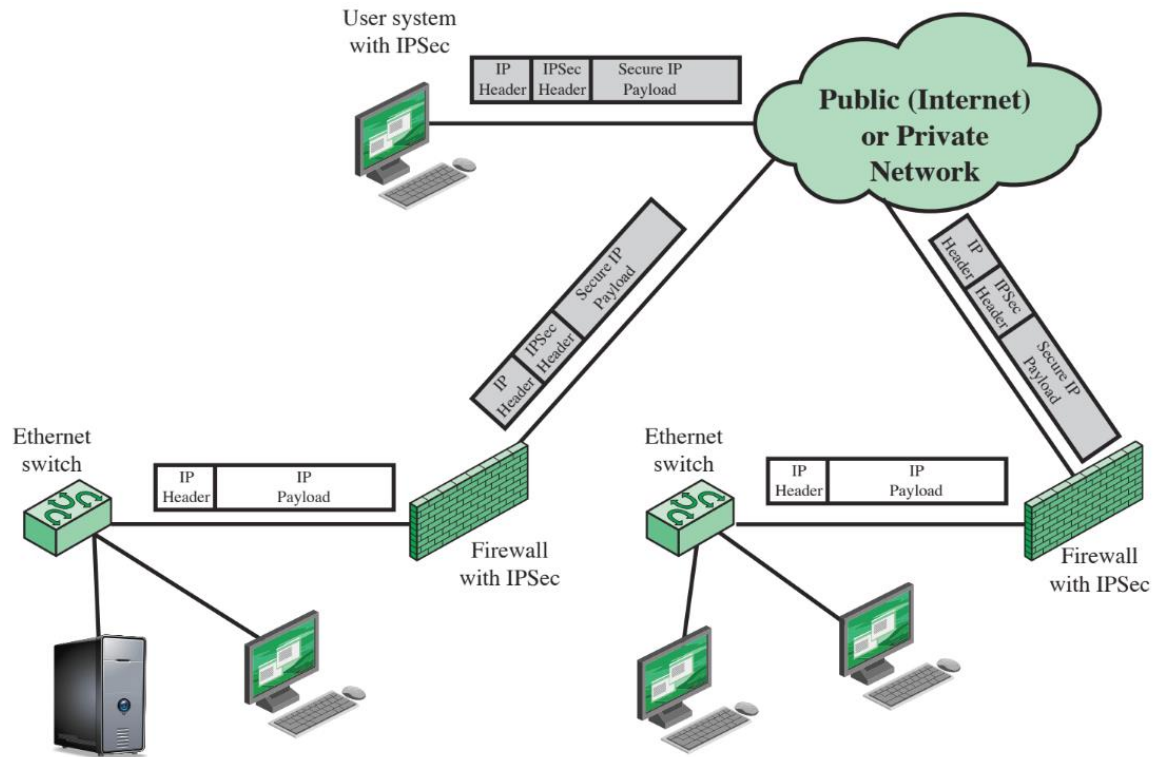


Figure 12.2 Example Firewall Configuration



**Figure 12.3 A VPN Security Scenario**

# Distributed firewall configuration

- Use both standalone and host-based firewalls w/ same administrative control.
- Admin. tools to configure firewalls on servers, workstations as well as host/personal firewalls
- Protect against internal attacks, and provide protection tailored to specific NEs and applications
- Establish internal and an external DMZs
- Adopt security monitoring, including log aggregation/analysis, traffic statistics, fine-grained monitoring of individual hosts

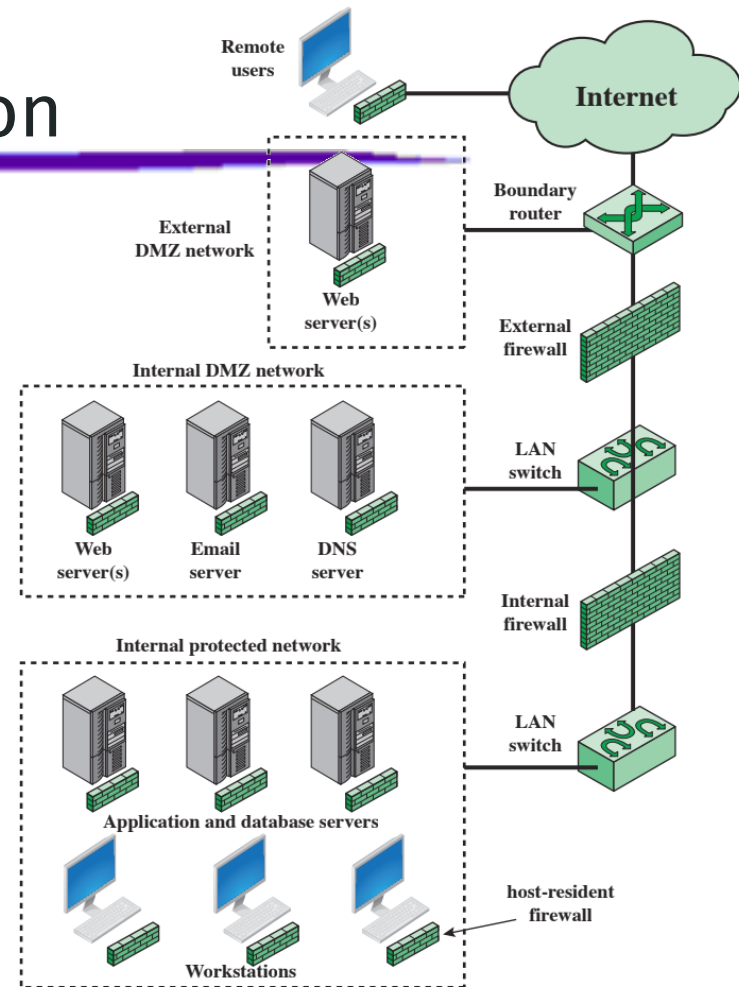


Figure 12.4 Example Distributed Firewall Configuration 15

# Problems with firewalls

---

- Can interfere with application use
- Limited protection against insiders – not necessarily true with security zoning
- Don't solve real problems: buggy implementations, bad protocols



# Intrusion detection

---

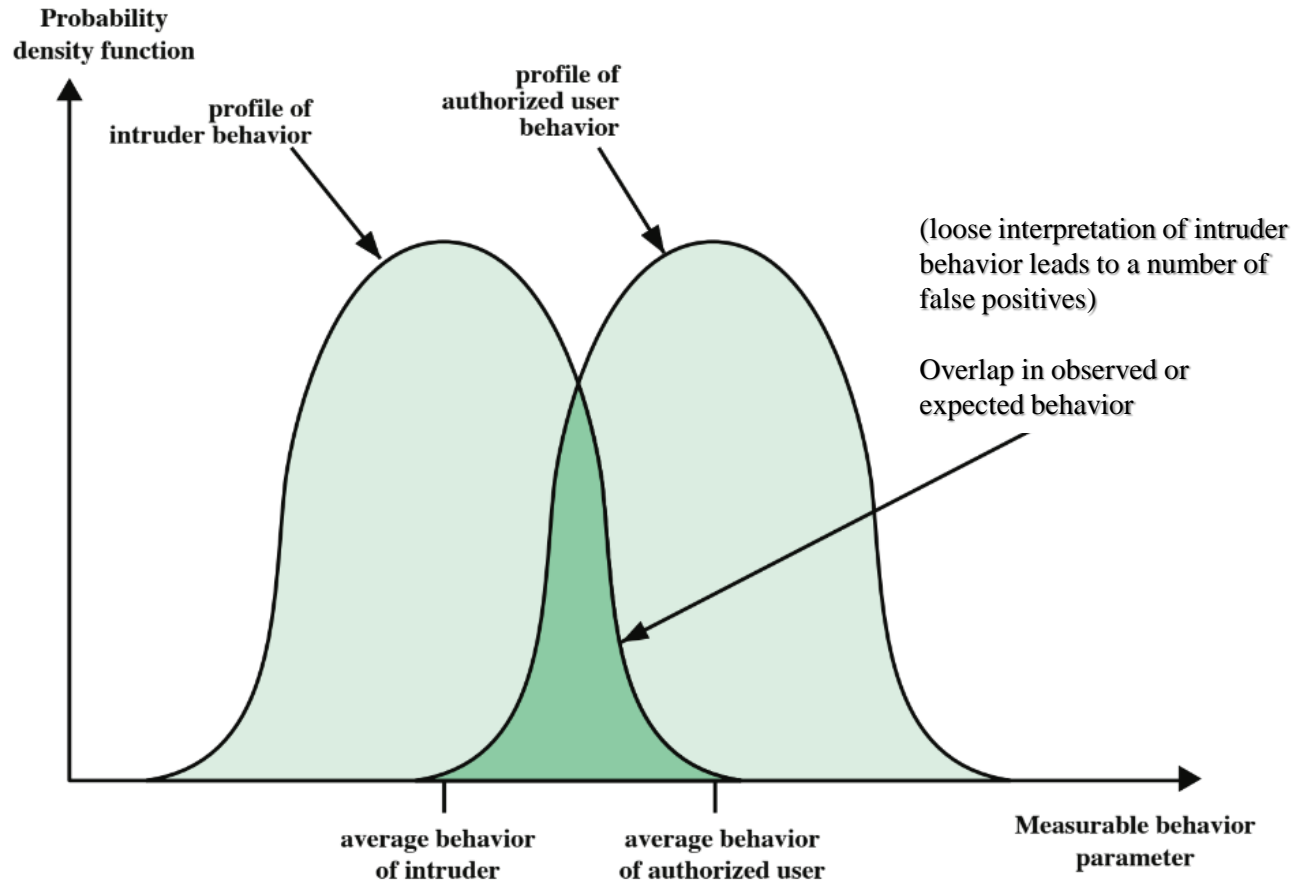
# Basic approach to intrusion detection

- Statistical anomaly detection: collect profile of authorized use/users, then apply statistical tests to ongoing use
- Rule-based detection: define a set of rules that correspond to a kind of intrusion, and check against that signature
- Use audit records (e.g. system to files) as input to intrusion detection system

# Base-rate fallacy

- Goal: detect most intrusions with fewest false positives
- If small percent of intrusions are detected, false security
- If too many false positives, managers will ignore alarms
- Base-rate fallacy: if number of intrusions are low compared to legitimate uses, false alarm rates will be high unless test is very discriminating

[Reference: Stallings, Appendix J]



**Figure 11.1 Profiles of Behavior of Intruders and Authorized Users**

# Statistical Anomaly Detection

- Threshold detection

- Involves counting the number of occurrences of a specific event type over an interval of time
- If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed
- By itself is a crude and ineffective detector of even moderately sophisticated attacks

- Profile-based

- Focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations
- A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert

# Audit Records

- Fundamental tool for intrusion detection

## Native audit records

Virtually all multiuser operating systems include accounting software that collects information on user activity

The advantage of using this information is that no additional collection software is needed

The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form

## Detection-specific audit records

A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system

One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems

The disadvantage is the extra overhead involved in having two accounting packages running on a machine

# Measures that may be used (Table 11.1, part 1)

Measure	Model	Type of Intrusion Detected
<b>Login and Session Activity</b>		
Login frequency by day and time	Mean and standard deviation	Intruders may be likely to log in during off-hours.
Frequency of login at different locations	Mean and standard deviation	Intruders may log in from a location that a particular user rarely or never uses.
Time since last login	Operational	Break-in on a "dead" account.
Elapsed time per session	Mean and standard deviation	Significant deviations might indicate masquerader.
Quantity of output to location	Mean and standard deviation	Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data.
Session resource utilization	Mean and standard deviation	Unusual processor or I/O levels could signal an intruder.
Password failures at login	Operational	Attempted break-in by password guessing.
Failures to login from specified terminals	Operational	Attempted break-in.

# Measures that may be used (Table 11.1, part 2)

Command or Program Execution Activity		
Execution frequency	Mean and standard deviation	May detect intruders, who are likely to use different commands, or a successful penetration by a legitimate user, who has gained access to privileged commands.
Program resource utilization	Mean and standard deviation	An abnormal value might suggest injection of a virus or Trojan horse, which performs side-effects that increase I/O or processor utilization.
Execution denials	Operational model	May detect penetration attempt by individual user who seeks higher privileges.

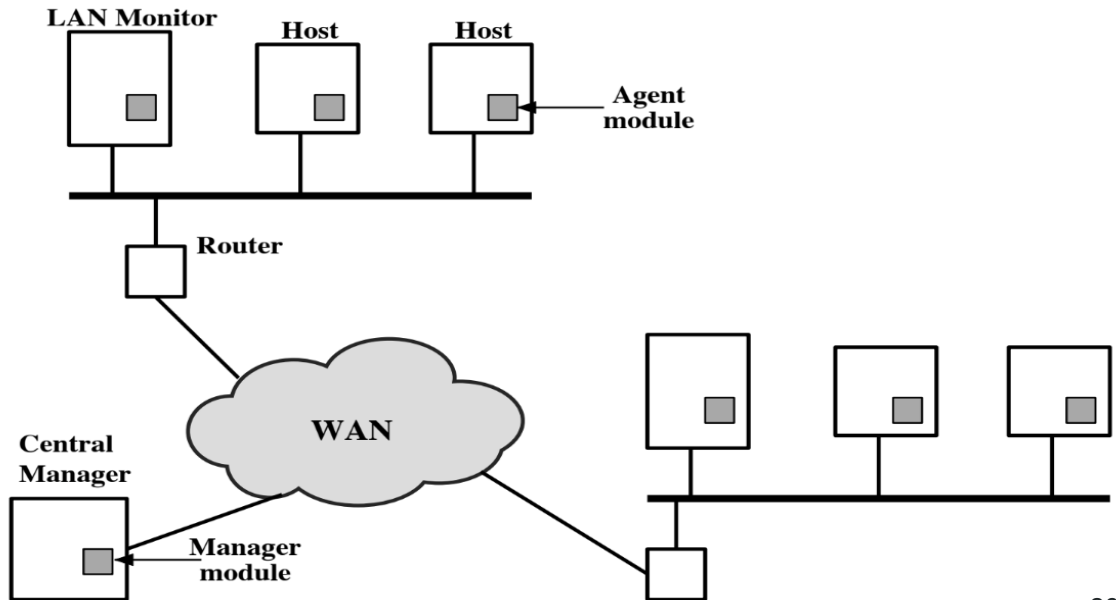


## Measures that may be used (Table 11.1, part 3)

	<b>File access activity</b>	
Read, write, create, delete frequency	Mean and standard deviation	Abnormalities for read and write access for individual users may signify masquerading or browsing.
Records read, written	Mean and standard deviation	Abnormality could signify an attempt to obtain sensitive data by inference and aggregation.
Failure count for read, write, create, delete	Operational	May detect users who persistently attempt to access unauthorized files.

# Distributed intrusion detection

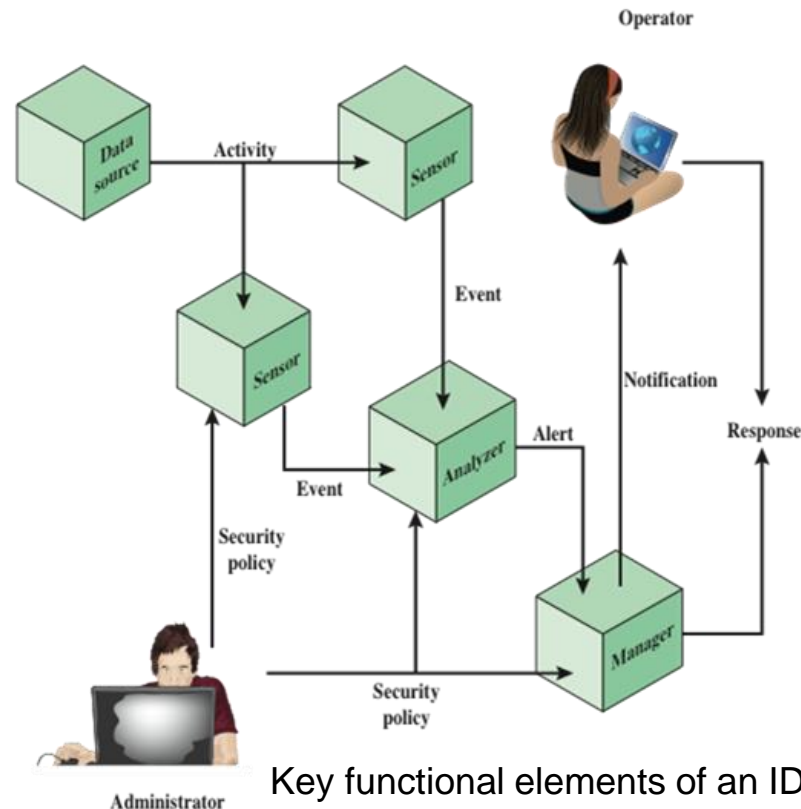
- Typical organization needs to defend distributed hosts supported by L2/L3 networks. Traditional single-node focused standalone facilities doesn't scale.
- More effective defense can be achieved by coordination among intrusion detection systems across the network
- Major design considerations:
  - Data collection and analysis points
  - Different audit record formats
  - Centralized/decentralized architecture



# Model for Intrusion Detection Message Exchange

IETF Intrusion Detection Working Group

- To facilitate the development of distributed intrusion detection systems
- Defining data formats and exchange procedures for sharing between response systems and to management systems
- Issued RFCs:
  - Intrusion Detection Message Exchange Requirements (RFC 4766), and the related message format (RFC 4765)
  - Intrusion Detection Exchange Protocol (RFC 4767)



Key functional elements of an IDS

# Honeypots

Decoy system to attract Intruders, that has no production value

- Divert attacker from real systems
- Identify attackers (real users have no reason to interact with honeypot)
- Collect information about attacker
- Keep attacker on Honeypots long enough to log, track, and respond

Must make sure that other systems in network are secure from activity on honeypot

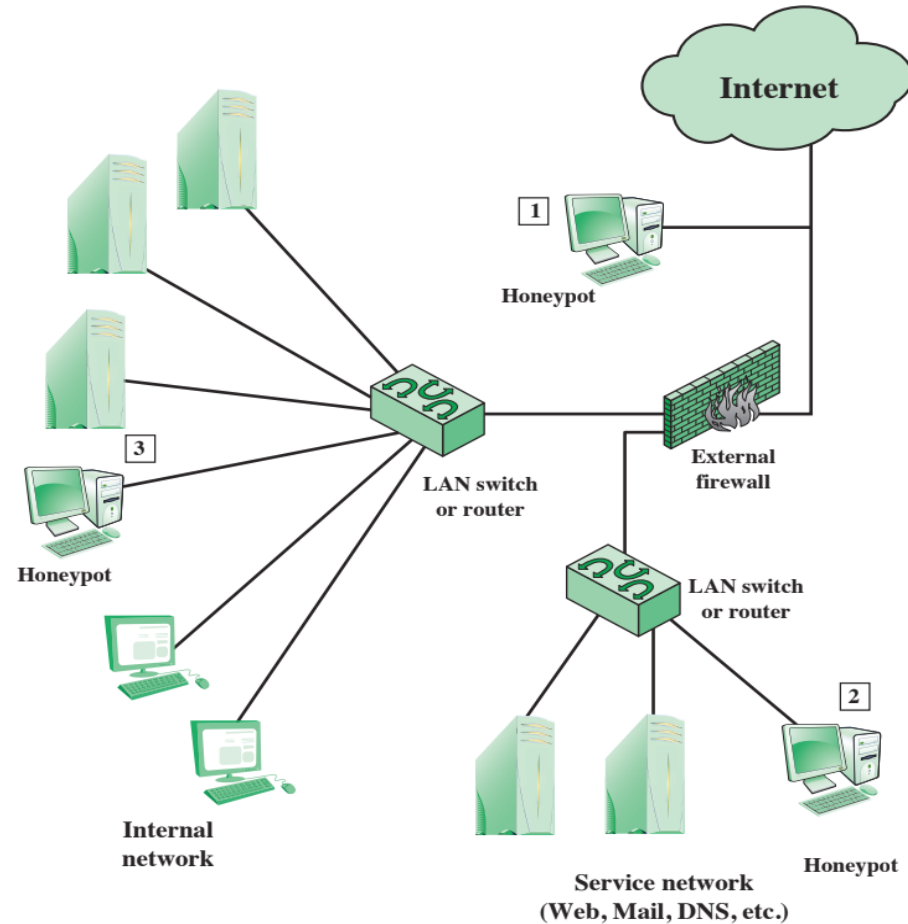


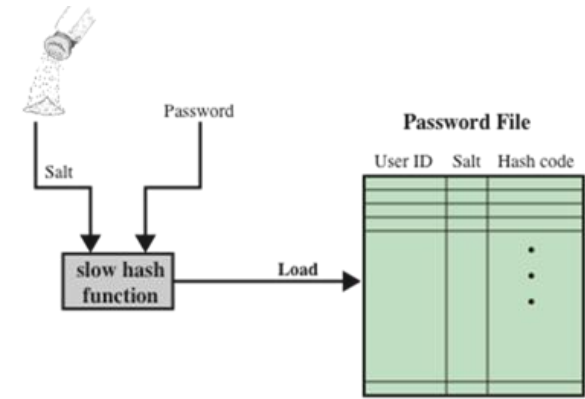
Figure 11.4 Example of Honeypot Deployment

# Password Management

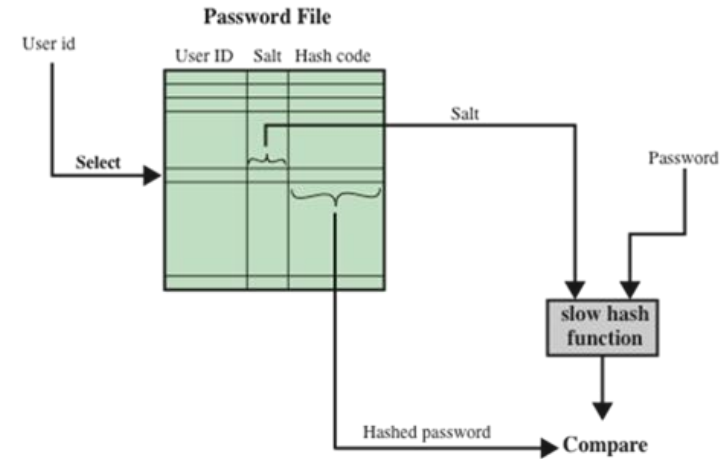
- Front line of defense against intruders
- Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password
  - Password serves to authenticate the ID of the individual logging on to the system
  - The ID provides security by:
    - Determining whether the user is authorized to gain access to a system
    - Determining the privileges accorded to the user
    - Used in discretionary access control

# Password security: use of hashed password & salt value

- Widely used by all UNIX variants as well as on a number of other operating systems
- A fixed-length salt value is combined with user password as inputs to hash algorithm
  - Salt value is related to the time when the password is created, or a random number (new implementation)
  - The hashing algorithm, which is designed to be slow to execute to thwart attacks, produces a hash code,
- Hashed password stored together with a plaintext copy of the salt in the password file for the corresponding user ID
- The hashed-password is checked when a user attempts to log on to a system ... ..



(a) Loading a new password



(b) Verifying a password

Example: denyhosts/fail2ban

---

# denyhosts/fail2ban intrusion prevention system

- Use system authentication logs, find IP address that is source of suspicious authentication attempts
- Install firewall rules to block that IP address



# fail2ban example

```
failregex = ^(__prefix_line)s(?:error: PAM: )?Authentication failure for .* from <HOST>\s*$
            ^(__prefix_line)s(?:error: PAM: )?User not known to the underlying authentication
module for .* from <HOST>\s*$
            ^(__prefix_line)sFailed(?:password|publickey) for .* from <HOST>(?: port \d*)?(?:
ssh\d*)?$$
            ^(__prefix_line)sROOT LOGIN REFUSED.* FROM <HOST>\s*$
            ^(__prefix_line)s[iI](?:illegal|nvalid) user .* from <HOST>\s*$
            ^(__prefix_line)sUser .+ from <HOST> not allowed because not listed in AllowUsers$
            ^(__prefix_line)s(?:pam_unix\(sshd:auth\):\s)?authentication failure; logname=\S*
uid=\S* euid=\S* tty=\S* ruser=\S* rhost=<HOST>(?:\s+user=.)?\s*$
            ^(__prefix_line)srefused connect from \S+ \(<HOST>\)\s*$
            ^(__prefix_line)sAddress <HOST> .* POSSIBLE BREAK-IN ATTEMPT!\s*$
            ^(__prefix_line)sUser .+ from <HOST> not allowed because none of user's groups are
listed in AllowGroups\s*$
```

# fail2ban example

```
Apr 10 21:42:11 witestlab sshd[17396]: Connection from 181.224.200.5 port 52766
Apr 10 21:42:11 witestlab sshd[17396]: Invalid user rosa from 181.224.200.5
Apr 10 21:42:11 witestlab sshd[17396]: input_userauth_request: invalid user rosa [preauth]
Apr 10 21:42:11 witestlab sshd[17396]: Received disconnect from 181.224.200.5: 11: Bye Bye [preauth]
Apr 10 21:42:12 witestlab sshd[17398]: Connection from 181.224.200.5 port 52948
Apr 10 21:42:12 witestlab sshd[17398]: Invalid user sa from 181.224.200.5
Apr 10 21:42:12 witestlab sshd[17398]: input_userauth_request: invalid user sa [preauth]
Apr 10 21:42:13 witestlab sshd[17398]: Received disconnect from 181.224.200.5: 11: Bye Bye [preauth]
Apr 10 21:42:14 witestlab sshd[17402]: Connection from 181.224.200.5 port 53312
Apr 10 21:42:15 witestlab sshd[17402]: Invalid user sai from 181.224.200.5
Apr 10 21:42:15 witestlab sshd[17402]: input_userauth_request: invalid user sai [preauth]
Apr 10 21:42:15 witestlab sshd[17402]: Received disconnect from 181.224.200.5: 11: Bye Bye [preauth]
Apr 10 21:42:15 witestlab sshd[17404]: Connection from 181.224.200.5 port 53510
Apr 10 21:42:16 witestlab sshd[17404]: Invalid user sam from 181.224.200.5
Apr 10 21:42:16 witestlab sshd[17404]: input_userauth_request: invalid user sam [preauth]
Apr 10 21:42:16 witestlab sshd[17404]: Received disconnect from 181.224.200.5: 11: Bye Bye [preauth]
```

# fail2ban example

Chain fail2ban-ip-blocklist (1 references)

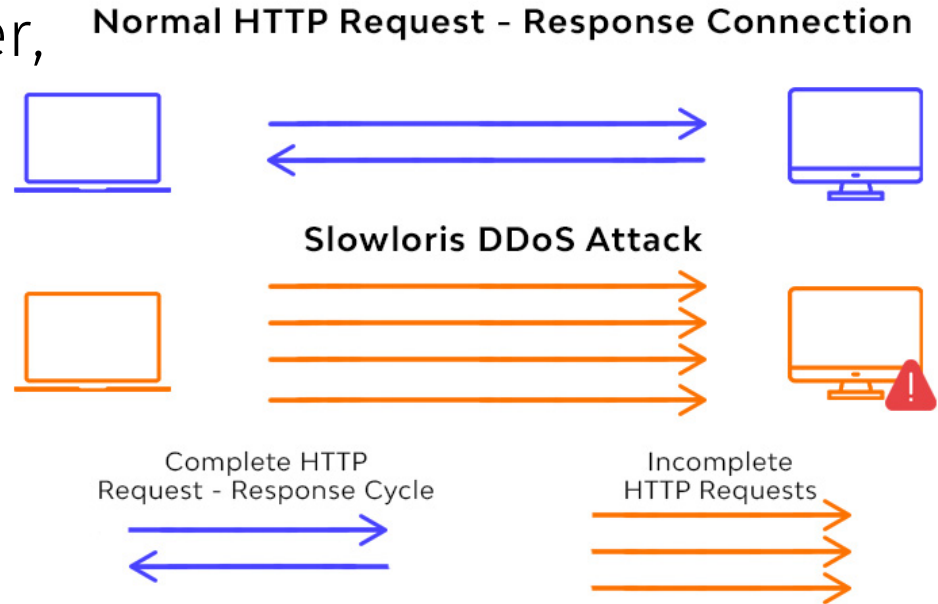
target	prot	opt	source	destination
DROP	all	--	<b>181.224.200.5</b>	0.0.0.0/0

# Example: slowloris defense

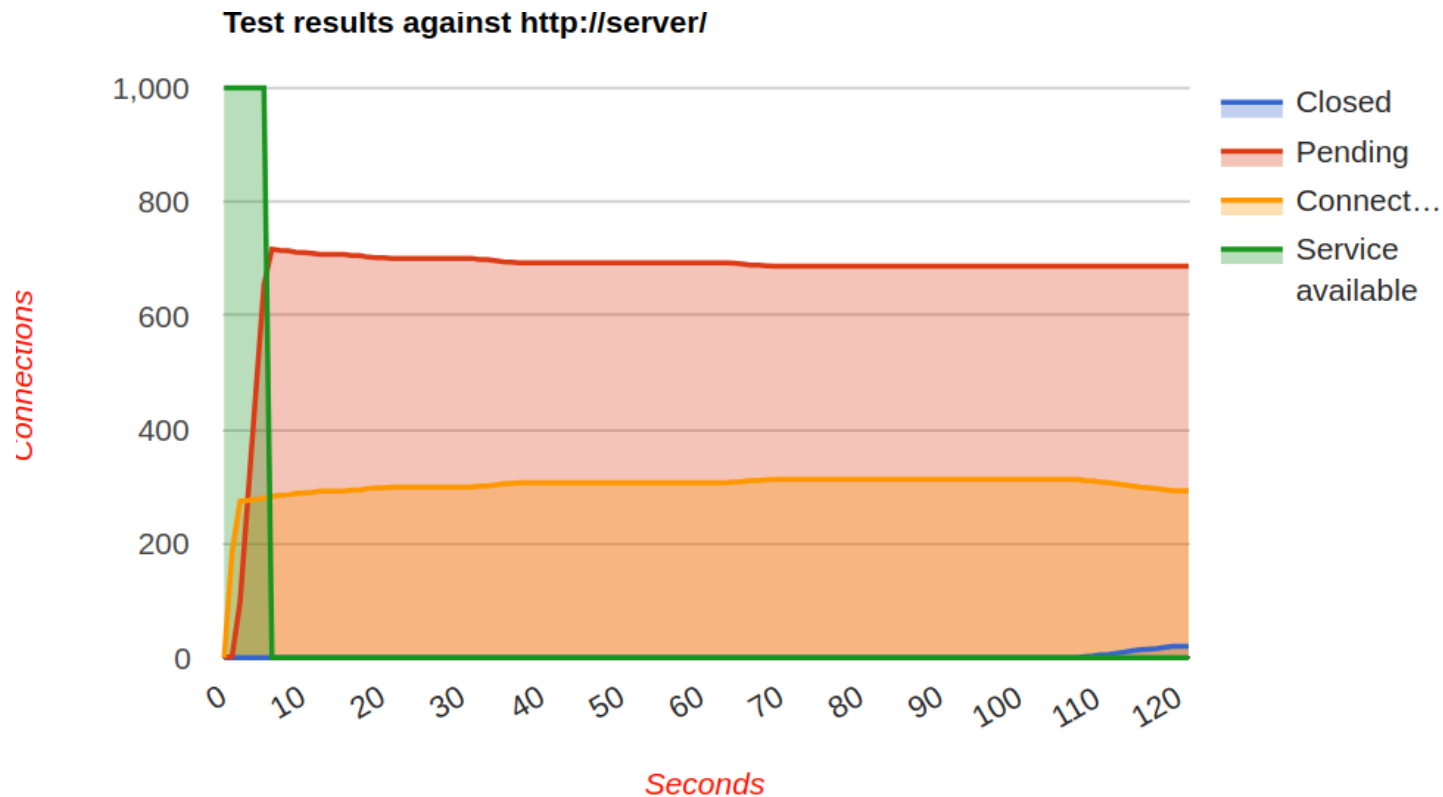
---

# Slowloris attack

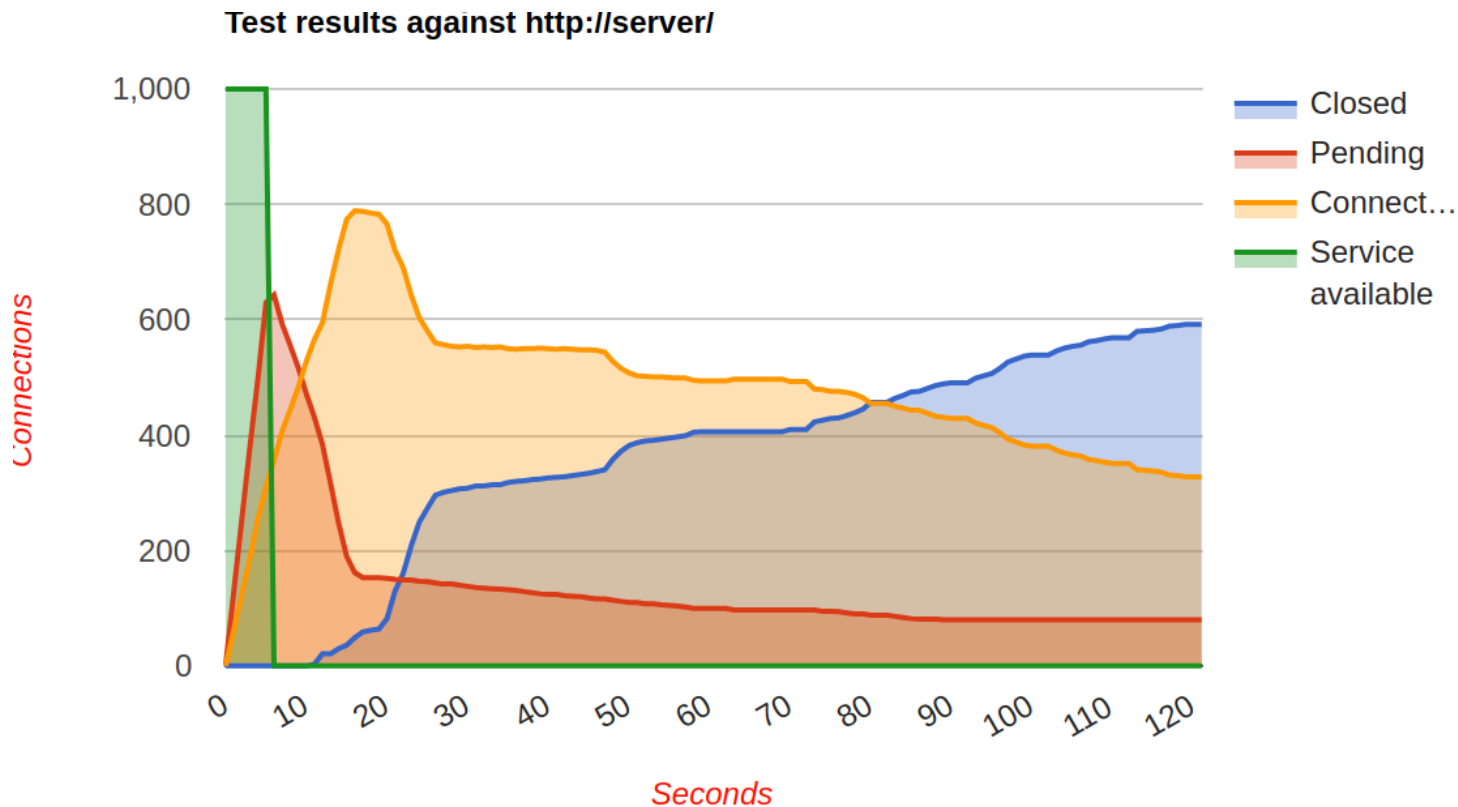
- Attacker opens many connections to web server, leaves open
- Web server has limited pool of threads
- Attacker exhausts thread pool, service is denied to legitimate users



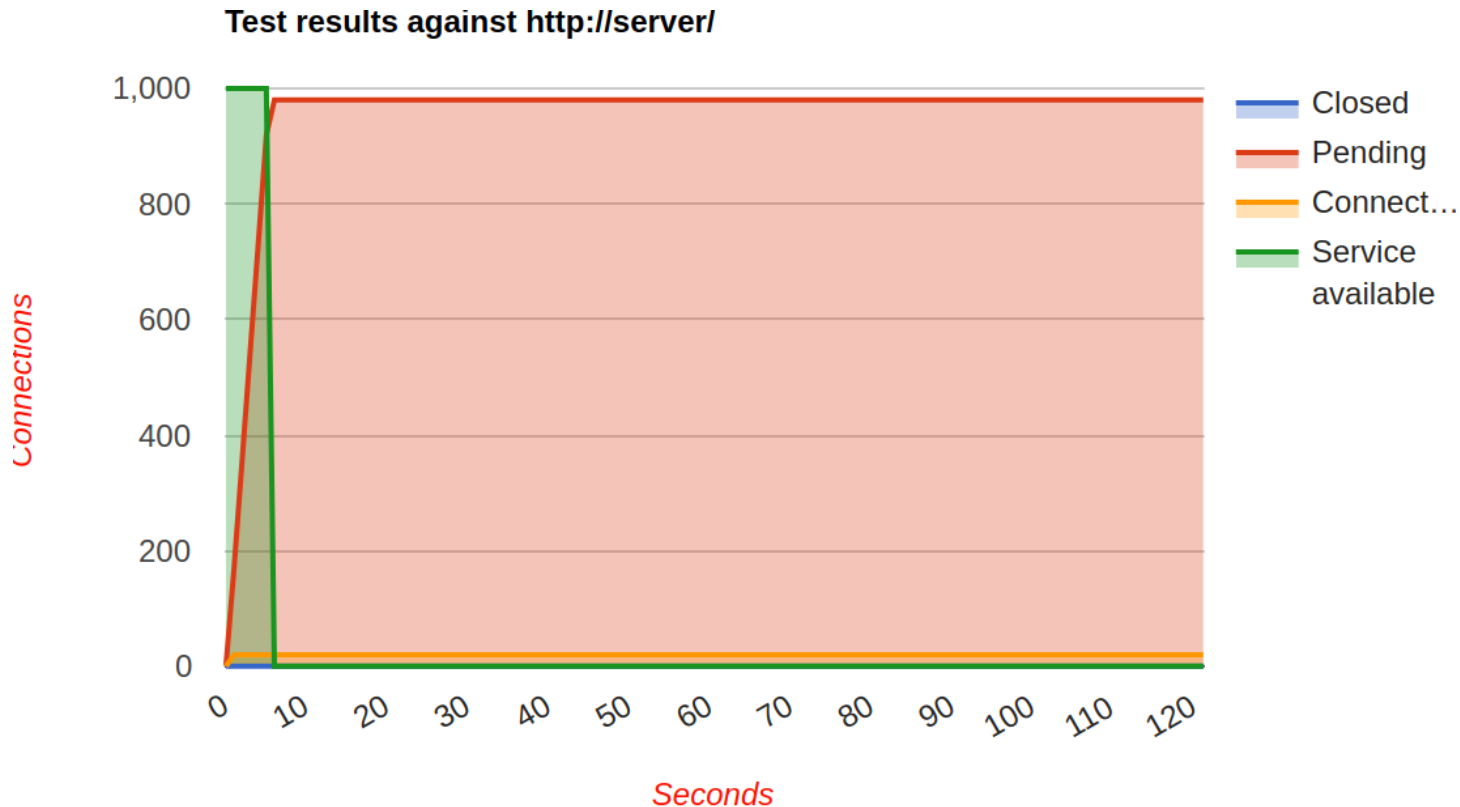
# System under attack



# Rate-limiting source



# Firewall to limit connections per source





# Alternative application design

