Network Security

AA 2015/2016

System hardening

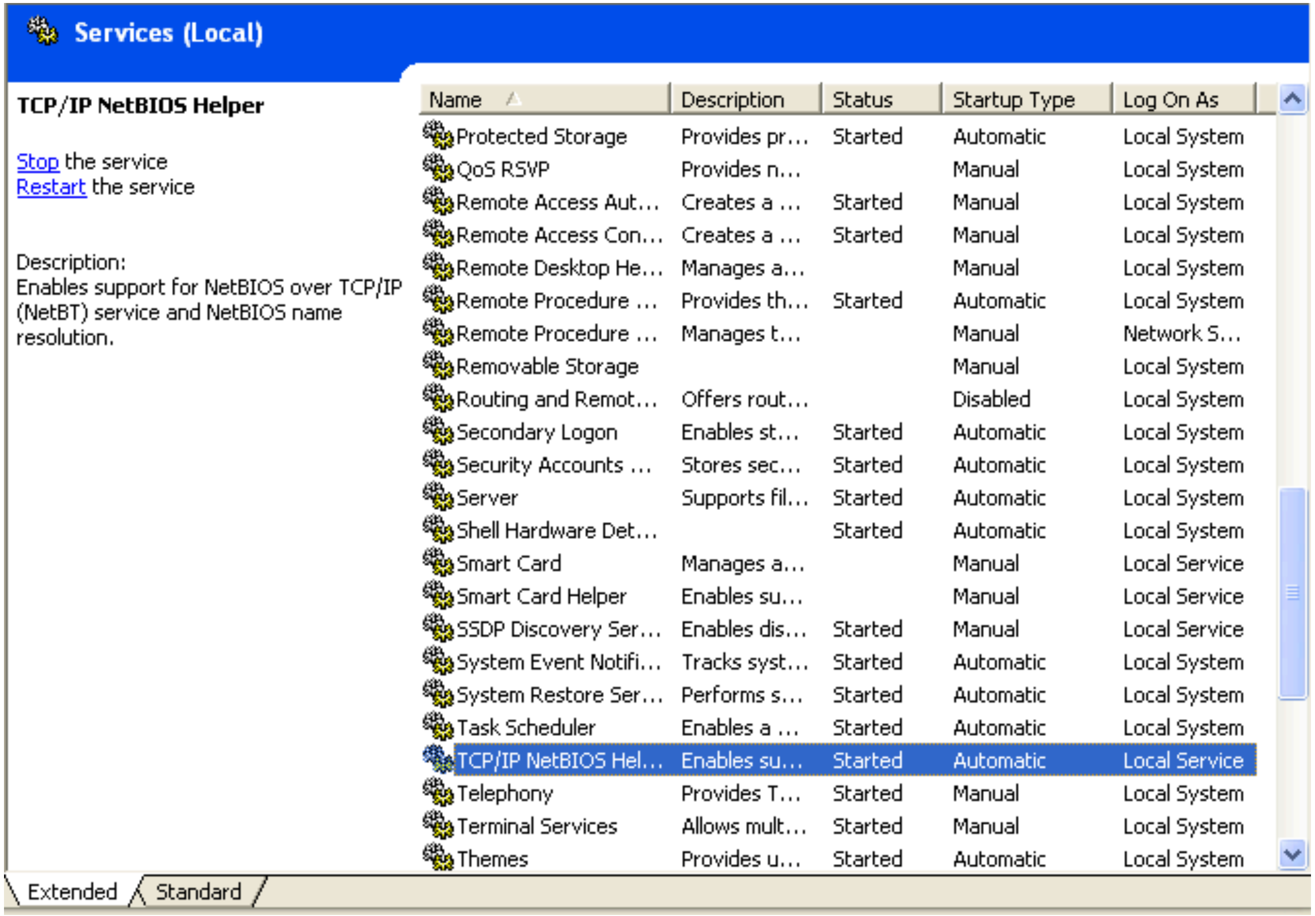
(Authentication, Firewalls)

Dr. Luca Allodi

Default configurations

- All systems have a default configuration
  - Personal computers, servers, mainframes,..
- Fresh installation of an operating system
  - Some can be configured at installation time
    - Still limited access to full configuration settings
    - e.g. linux distro typically allows to select packets but not all packet functionalities
- Default services
  - DHCP, RCP, NetBIOS, ..
  - SSH, VNC, ..
  - Web servers, remote interfaces
- → Default configuration satisfies vast majority of user needs

Example of default configuration



Services (Local)

TCP/IP NetBIOS Helper

Stop the service
Restart the service

Description:
Enables support for NetBIOS over TCP/IP (NetBT) service and NetBIOS name resolution.

Name	Description	Status	Startup Type	Log On As
Protected Storage	Provides pr…	Started	Automatic	Local System
QoS RSVP	Provides n…		Manual	Local System
Remote Access Aut…	Creates a …	Started	Manual	Local System
Remote Access Con…	Creates a …	Started	Manual	Local System
Remote Desktop He…	Manages a…		Manual	Local System
Remote Procedure …	Provides th…	Started	Automatic	Local System
Remote Procedure …	Manages t…		Manual	Network S…
Removable Storage			Manual	Local System
Routing and Remot…	Offers rout…		Disabled	Local System
Secondary Logon	Enables st…	Started	Automatic	Local System
Security Accounts …	Stores sec…	Started	Automatic	Local System
Server	Supports fil…	Started	Automatic	Local System
Shell Hardware Det…		Started	Automatic	Local System
Smart Card	Manages a…		Manual	Local Service
Smart Card Helper	Enables su…		Manual	Local Service
SSDP Discovery Ser…	Enables dis…	Started	Manual	Local Service
System Event Notifi…	Tracks syst…	Started	Automatic	Local System
System Restore Ser…	Performs s…	Started	Automatic	Local System
Task Scheduler	Enables a …	Started	Automatic	Local System
TCP/IP NetBIOS Hel…	Enables su…	Started	Automatic	Local Service
Telephony	Provides T…	Started	Manual	Local System
Terminal Services	Allows mult…	Started	Manual	Local System
Themes	Provides u…	Started	Automatic	Local System

System hardening

- System hardening is the process by which a system's configuration is tuned to improve its security without compromising its functionality
    - The 100% secure system is one that is turned off
- Sys hardening process takes into account
    - System functionality → what is the role of that system?
        - Home computer
        - File server
        - Web server
        - General purpose server
    - System security → how can the security of the system be improved?
        - Minimise the attack surface of the system

Attack surfaces

- An attack surface is the set of system resources that are exposed to the attacker
  - Weak passwords
  - Software vulnerabilities
  - Misconfigurations
  - Services listening on the network
  - Inaccurate access control
  - …
- Golden rule of information security
  - "Minimality principle" → no user and no system component or process should be authorised or compiled to perform actions that are not strictly necessary for their normal operation
    - aka "If it's not there you can't brake it"

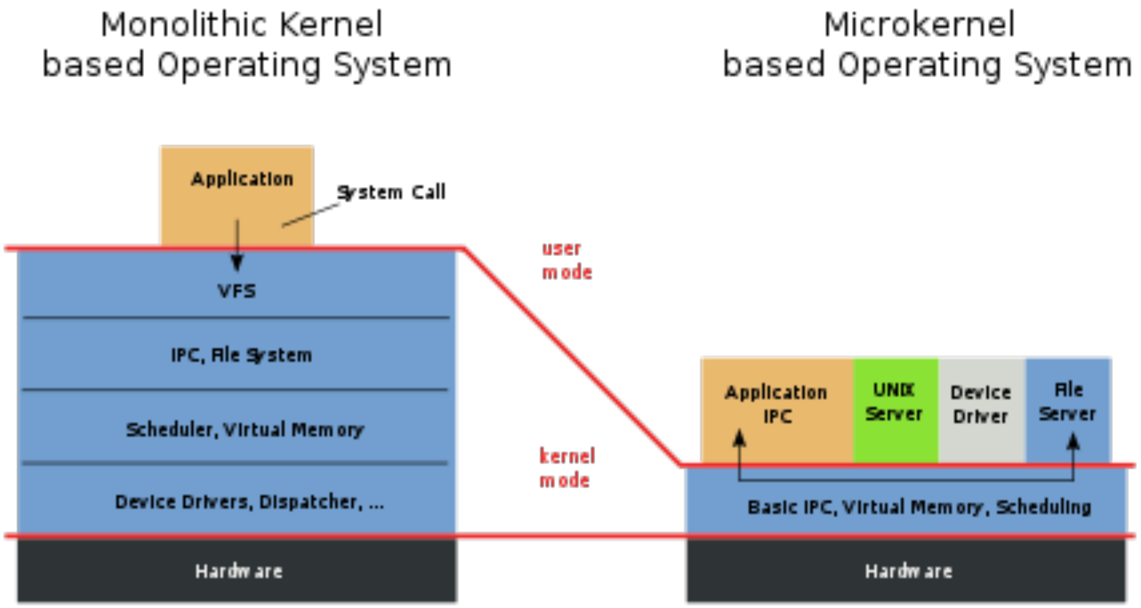
The minimality principle

- Can be applied at both system users and processes
- **A system** should be configured such that it does not embed or enable functionalities that are not needed for normal operation
  - example: microkernel → Liedtke's minimality principle:
    - *A concept is tolerated inside the microkernel only if moving it outside the kernel, i.e., permitting competing implementations, would prevent the implementation of the system's required functionality.*
- **A user** should be authorised to only access and modify resources that are necessary for their normal operation
  - If user is NOT authorised, they will NOT be able to accomplish their tasks

Minimal system configuration

- Heavily depends on system functionality
- There is not one "Best secure configuration" that fits all systems
- Best solution depends on a number of design/environment variables
- Example:
    - What's the system designed for?
        - General computation server
    - Does it need local/remote access? → remote
        - If remote only, does it really need physical input interfaces? → no, take keyboards out
        - Need for multiple users? → yes, one admin and 20 students
    - What services should be accessible and from where?
        - Can devise environment conditions to regulate access? →yes, remote access only allowed from local area network → all activities logged → input devices disabled (e.g. no USB mount service)
- Default operating system installation often has several unnecessary functionalities enabled
    - Rely on documentation to decide what's necessary and what's not
        - You remove something useful → brick the system
    - Compile your own kernel (when possible) → can be done as a trial-and-error by restoring previous kernel if something goes wrong

Example of minimal design for security: Microkernel structure


Monolithic Kernel based Operating System
Microkernel based Operating System
Application
System Call
VFS
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
user mode
kernel mode
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware

Minimal user privileges

- User should not be allowed to perform more actions on the system than necessary for their operation
- Typically implemented via **user authentication**
- Common policy requirement: restrict the behavior of a user
- To permit different users to do different things, we need a way to identify or distinguish between users
  - Identification mechanisms to indicate identity
  - Authentication mechanisms to validate identity

User Authentication

User Authentication

- is the process of verifying an identity claimed by or for a system entity
- fundamental security building block
  - basis of access control → user accountability
- has two steps:
  - identification – presenting identifier to the security system
  - verification – presenting information that corroborates the binding between entity (person) and identifier
- **Final goal → link physical user of the system with their representation in the system**
  - Typically done through the existence of a "secret" that only the physical person corresponding to that system representation can know/possess/derive
- distinct from message authentication

Means of User Authentication

- four means of authenticating user's identity
- based one something the individual
  - **knows** - e.g. password, PIN, graphical password
  - **possesses** - e.g. key, token, smartcard
  - **is (static biometrics)** - e.g. fingerprint, retina
  - **does (dynamic biometrics)** - e.g. voice, sign
- can use alone or combined
- all can provide user authentication
- all have issues

Something you know: Password Authentication

- widely used user authentication method
  - user provides name/login and password
  - system compares password with that saved for specified login
- authenticates ID of user logging and
  - that the user is authorized to access system
  - determines the user's privileges
- Sequence of characters
  - Examples: 10 digits, a string of letters, *etc*.
    - Luca, Lyk4, !Luca!, !£L^y]k@#4!, ..
  - Generated randomly, by user, by computer with user input
    - 432432k-12312j-sdfjs1-24554g ← user-generated "random" string
- Sequence of words
  - Examples: pass-phrases
    - Luca started the Network Security course on the fiftheenth of February

Problem: Password Storage

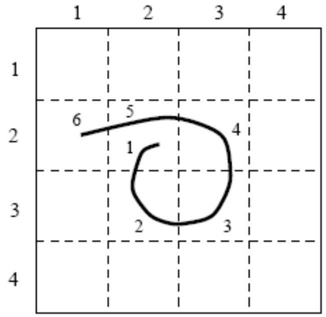
- Store as cleartext
  - If password file compromised, *all* passwords revealed
- Encrypt file
  - Need to have decryption, encryption keys in memory
  - Reduces to previous problem
- Store one-way hash of password
  - If file read, attacker must still guess passwords or invert the hash
- Hashed passwords
  - Password is concatenated with a random salt → store H(salt+password)
  - Avoids problem whereby same passwords have same hash value

Password Aging

- "Frequently" change passwords decreases attack surfaces
  - Lower probability of having a breach
  - Less time for attacker to crack hash file
- Force users to change passwords after some time has expired
- Users will have to create and remember more passwords for one account
  - How do you force users not to re-use passwords?
    - Record hashes of previous passwords
    - Block changes for a period of time
  - Give users time to think of good passwords
    - Don't force them to change before they can log in
    - Warn them of expiration days in advance
- Balance between security and usability

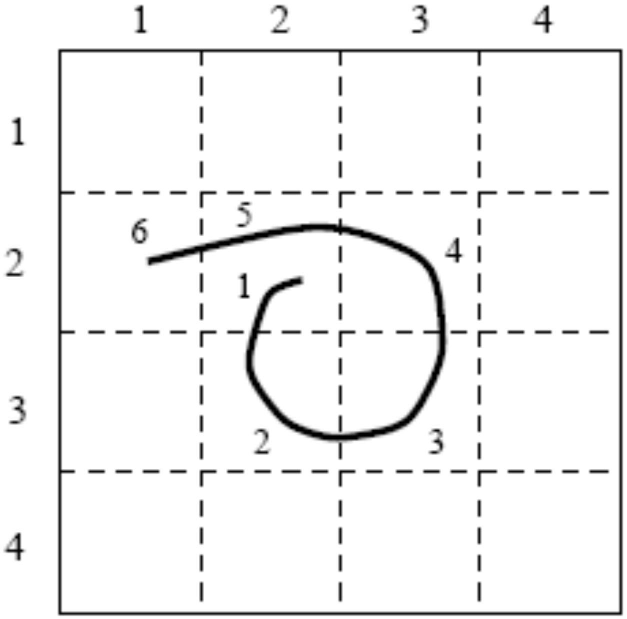
Draw-A-Secret (DAS) Scheme

- Password is picture drawn on a grid



- Users **freed** from having to remember **alphanumeric string**
- Pros:
  - Easier to remember
  - Low error rates
- Cons:
  - Adjacent coordinates more likely to be used in sequence
  - On touch screens could be easy to retrieve combination

Draw-A-Secret (DAS) Scheme

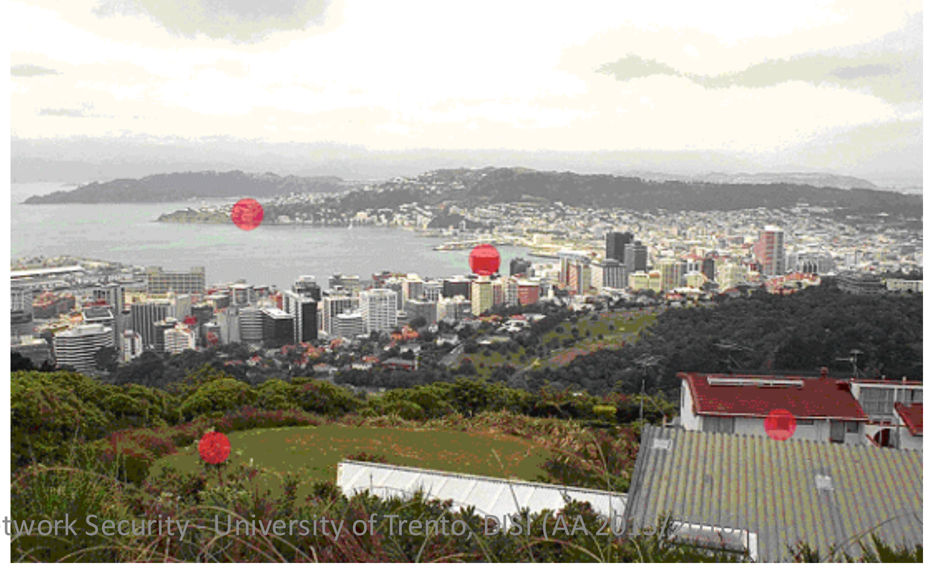


(2,2) (3,2) (3,3) (2,3) (2,2) (2,1) (5,5)

(5,5) is pen-up indicator

Another graphical password scheme

- To login, user is required to click within the circled red regions (chosen when created the password) in this picture.
    - The choice for the four regions is arbitrary → user preference
- Known since the mid 1990s,
- "Graphical Passwords" →
  http://rutgersscholar.rutgers.edu/volume04/sobrbirg/sobrbirg.htm
- Drawbacks
    - Shoulder surfing → the attacker can easily see the combination on screen
    - Unclear: easy to change for the user?



Something you have: Token Authentication

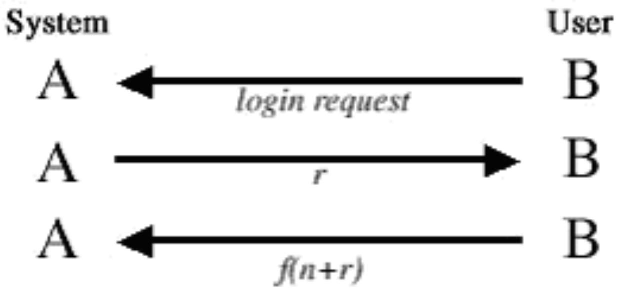
- Tokens - objects that a user possesses to authenticate, e.g.:
  - embossed card
  - magnetic stripe card
  - memory card
  - smartcard

Hardware Support

- Token-based
  - Used to compute response to challenge (see challenge-response next)
    - May encipher or hash challenge
    - May require PIN from user
  - Temporal password generation
    - Every minute (or so) different number shown
      - Server knows what number to expect and when
    - User enters number and fixed password

Challenge-response

- The attacker (MitM) can not observe actual value, but only the challenge and the response
  - Can not reverse function that computes the response



n → shared secret
r → challenge
f(n+r) → response

- f() can be any one-way function
  - Hash → computation by system
  - image random operations (rotation, shifts, ..) → computation by human
- Can be used to prevent *shoulder surfing* → even if attacker sees current value, can not predict next valid r

One-Time Passwords

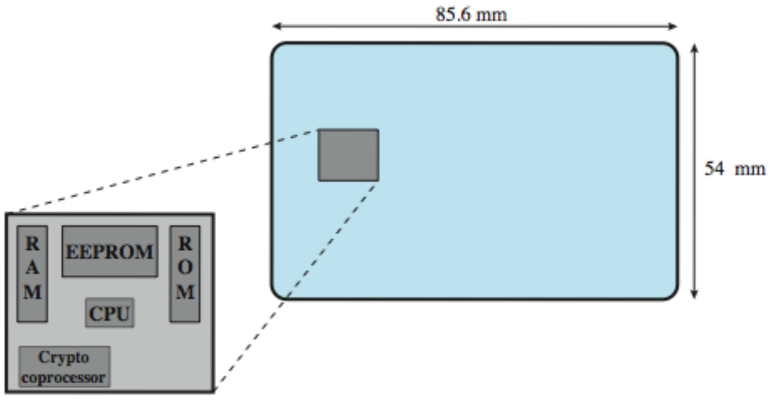
- Password that can be used exactly *once*
  - Often generated by a token
    - Other means include text messages, phone applications, ..
  - After use, it is immediately invalidated
- Challenge-response mechanism
  - Depends on implementation
  - Most common is time-syncronization → token and server have sync'd clock → will generate same number r at a given time
    - r=f(shared_secret, time)
    - time is challenge
    - r is response = one-time password
  - UserID + PIN + r → user authentication
- Problems
  - Synchronization of user, system
  - Generation of good random passwords
  - Password distribution problem

Memory Card

- store but do not process data
- magnetic stripe card, e.g. bank card
- electronic memory card
- used alone for physical access
- with password/PIN for computer use
- drawbacks of memory cards include:
  - need special reader → a common card reader can copy/overwrite security code
  - loss of token
  - user dissatisfaction for computer use

Smartcards

- Have own processor, memory, I/O ports
  - wired or wireless access by reader
  - may have crypto co-processor
  - ROM, EEPROM, RAM memory
- Execute protocol to authenticate with reader/computer
  - also have USB dongles
- Can be used to store
  - enc keys (GPG)
  - Certificates (Bitlocker, Firefox)
- Tamper-resistant



typical chip layout

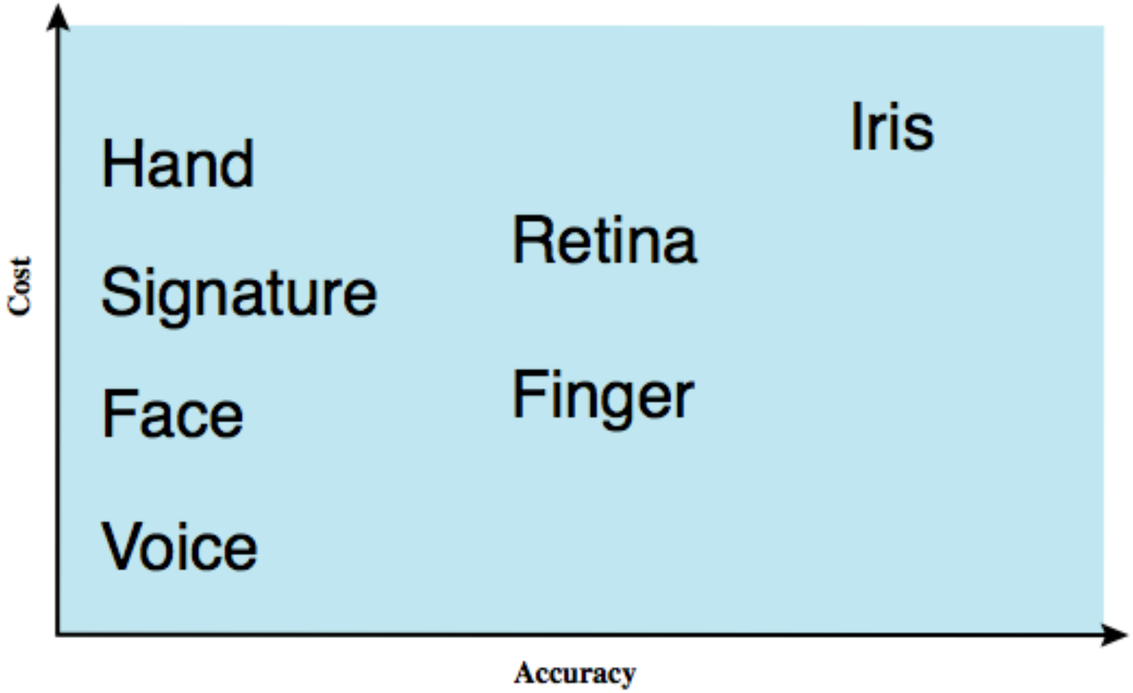
Something you are/do: Biometrics for Authentication

- *A biometric is a physiological or behavioral characteristic of a human being that can distinguish one person from another and that can be used for identification or verification of identity.”*

- Biometric applications available today are categorized into 2 types:
  - Physiological (static): Iris, Fingerprints, Hand, Retinal and Face recognition
  - Behavioral (dynamic): Voice, Typing pattern, Hand Signature, gesture, gait

Biometric Authentication

- authenticate user based on one of their physical characteristics



Physiological

- Automated measurement of biological features that identify a person
  - Fingerprints: optical or electrical techniques
    - Several different types: arch, whorl, loop, ..
    - Maps fingerprint into a graph, then compares with database
    - Measurements imprecise, so approximate matching algorithms used

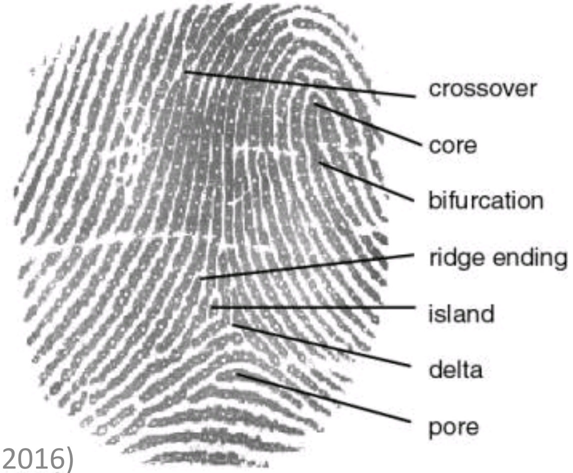
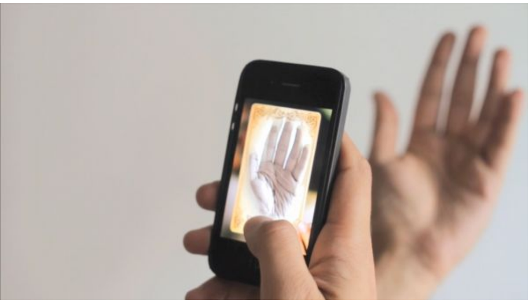


Figure 1

Physiological

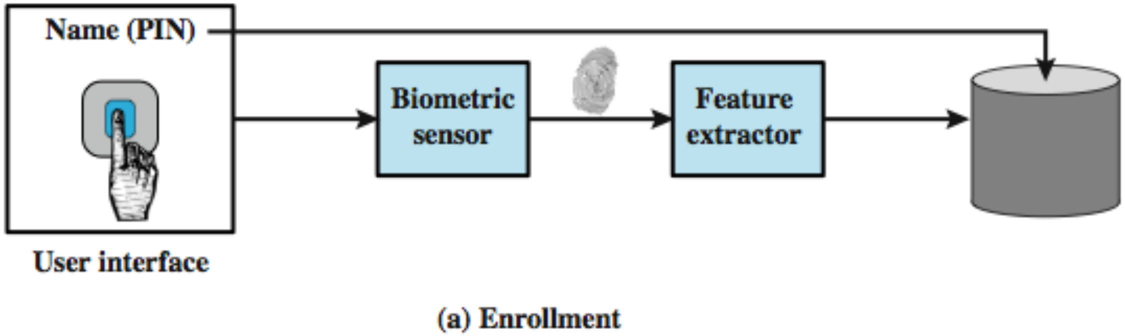


- Can use several other characteristics
  - Eyes: patterns in irises unique
    - Measure patterns, determine if differences are random; or correlate images using statistical tests
  - Palm recognition: believed to be unique
    - Not very robust and easy to forge if readers are cheap
    - Statistical tests used
  - Faces: image, or specific characteristics like distance from nose to chin
    - Lighting, view of face, other noise can hinder this
    - Issue with face recognition

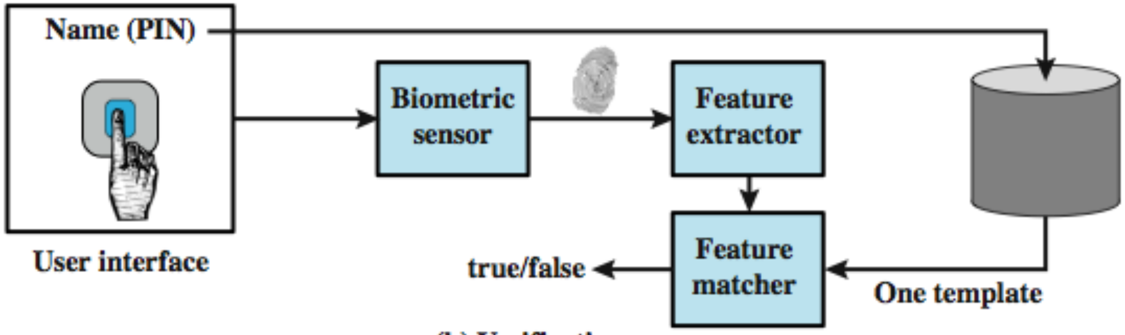
Behavioural

- Voices: speaker verification or recognition
  - Verification, recognition: uses statistical techniques to test hypothesis that speaker is who is claimed (speaker dependent), and recognize answer (content)
- Hand signature recognition
  - Speed, velocity, pressure
  - High user acceptance

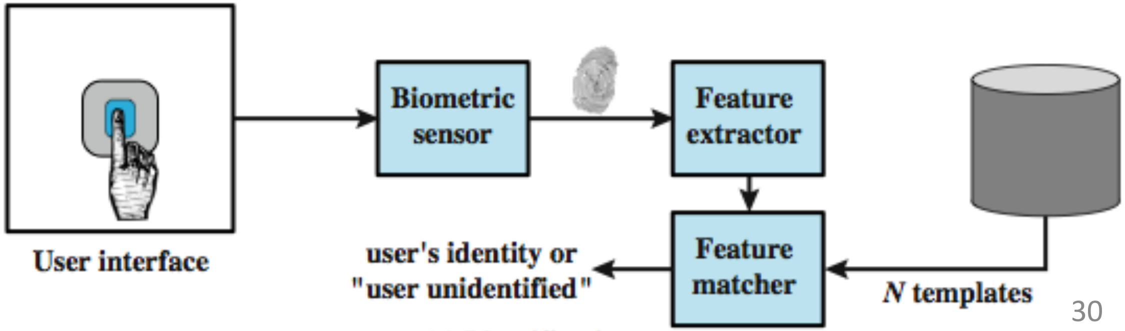
Operation of a Biometric System



(a) Enrollment



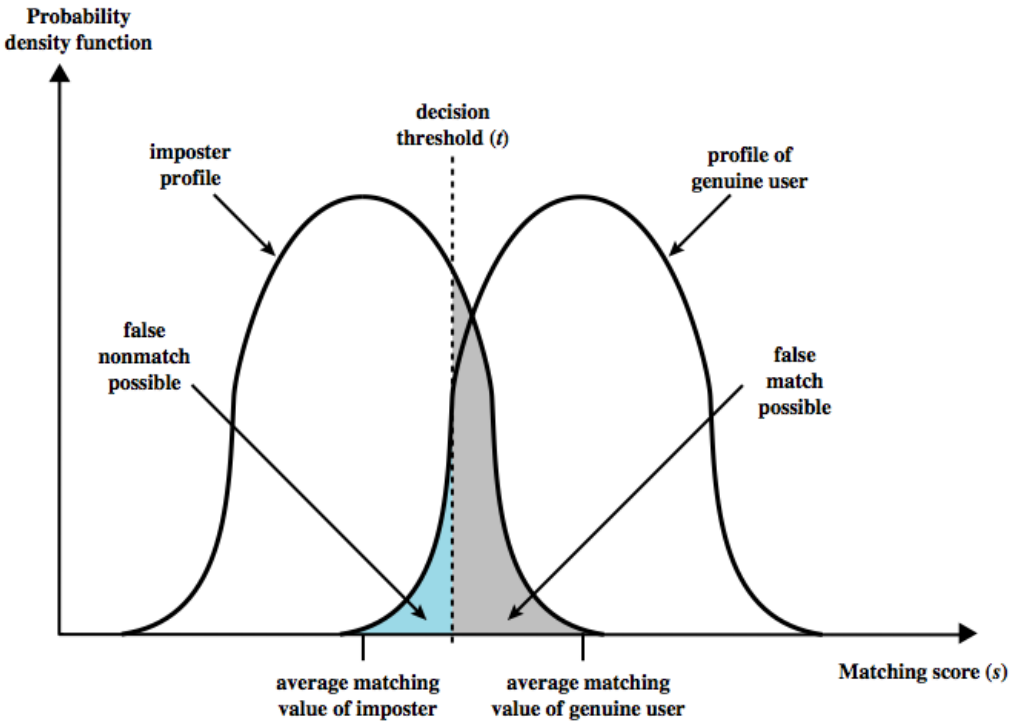
(b) Verification



(c) Identification

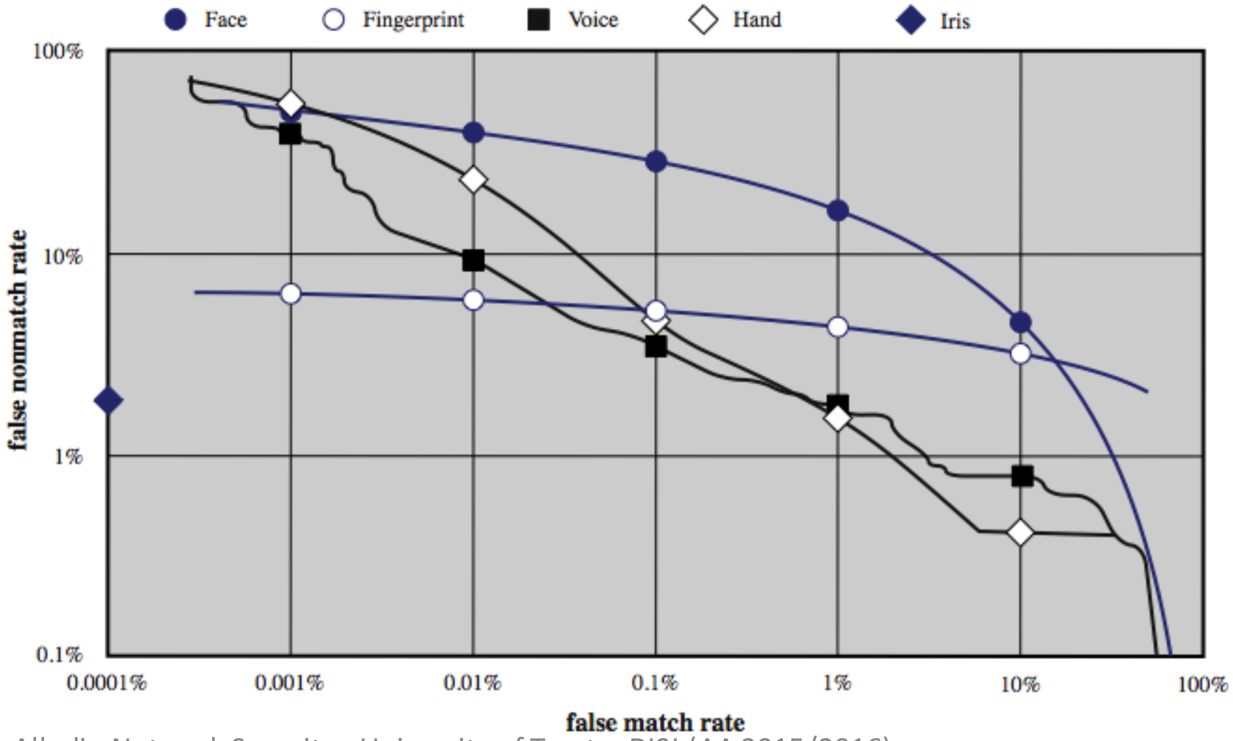
Biometric Accuracy

- never get identical templates
- problems of false match / false non-match



Biometric Accuracy

- can plot characteristic curve (ROC)
- pick threshold balancing error rates

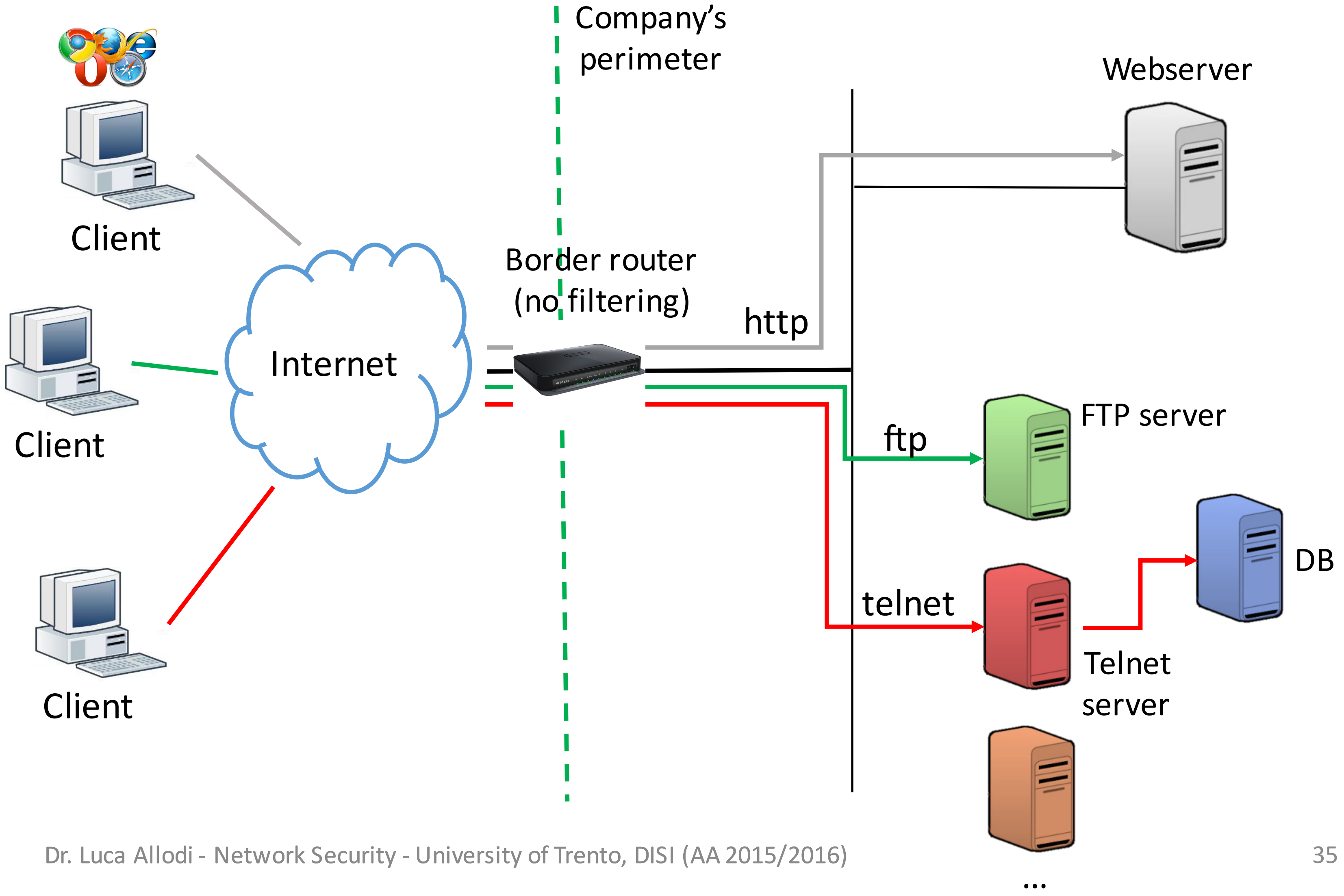


Firewalls

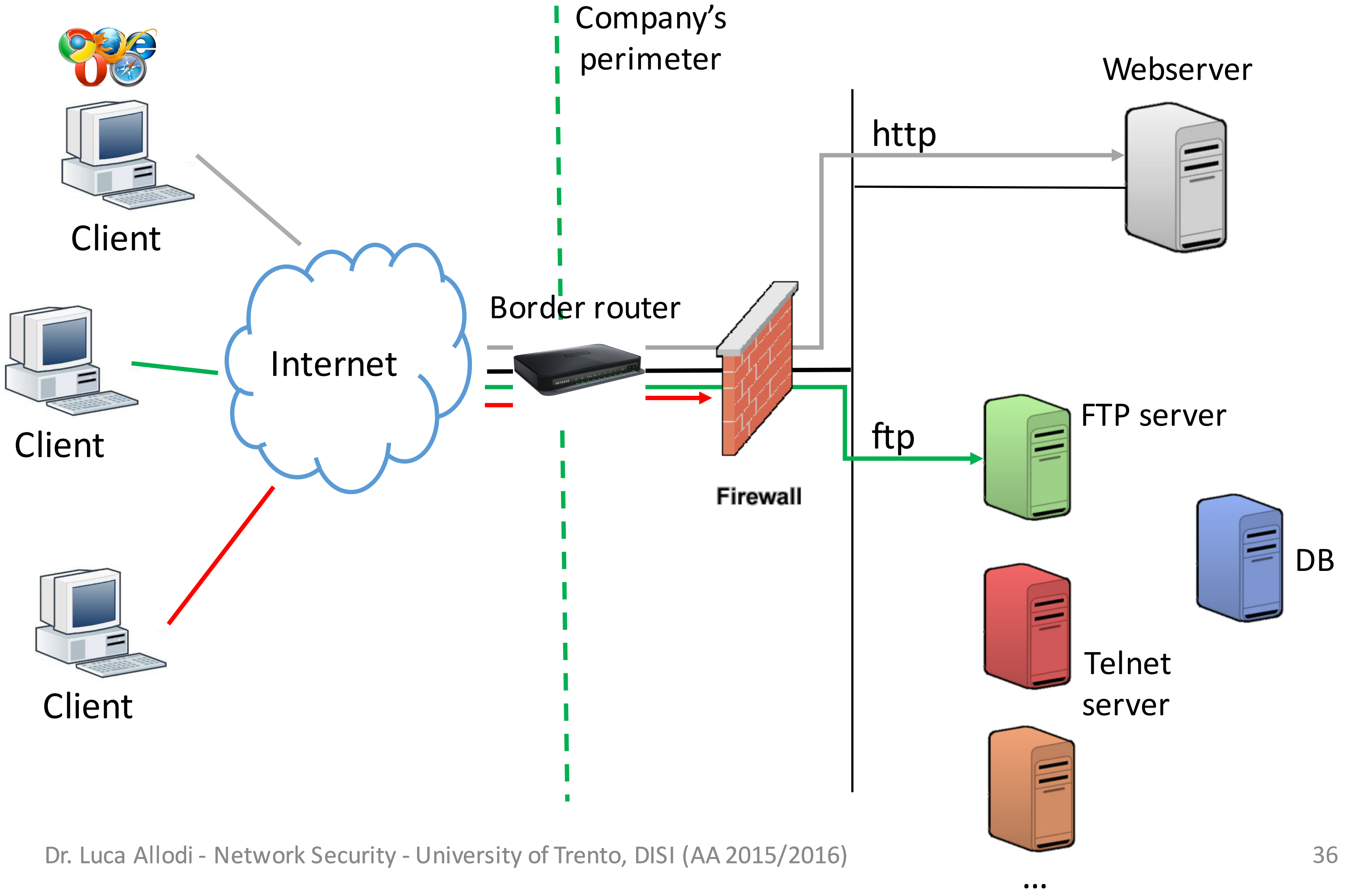
Firewalls for system minimality

- A system’s minimal configuration may still have a higher attack surface than necessary
  - e.g. SSH is necessary for remote operation on server
  - However, SSH logins may only be allowed only if from an internal IP address
    - Additional network measures to minimise attack surface
- Firewalls are perimetral network components that filter incoming (outgoing) traffic from (to) the network
  - Physical or software firewalls

No perimetral defense

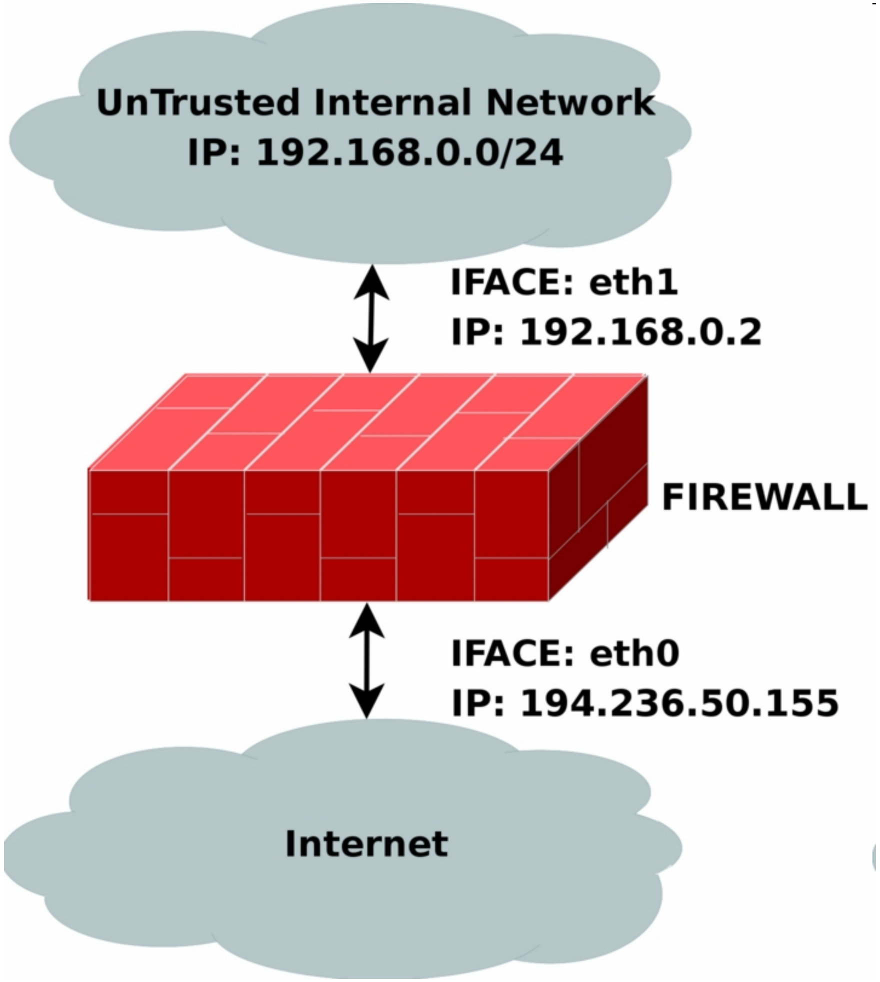


Perimetral defense



Networking with a firewall

- Internal network can be treated as untrusted
  - Do not trust outgoing traffic
  - Connections to remote servers can be regulated
    - E.g. remote storage services could be used to exfiltrate data from an organisation
- Firewalls have at least two network interfaces
  - One facing the external network
    - Or the router
    - This depends on firewall placement w.r.t border router
  - One facing internally
- More interfaces are possible if the firewall sits at the border with three or more networks



Firewall Characteristics

- Design goals
  - All traffic from inside or outside must pass through the firewall (physically blocking all access to the local network except via the firewall)
  - Only authorized traffic (defined by the local security policy) will be allowed to pass
  - The firewall itself is immune to penetration (use of a trusted system with a secure operating system)

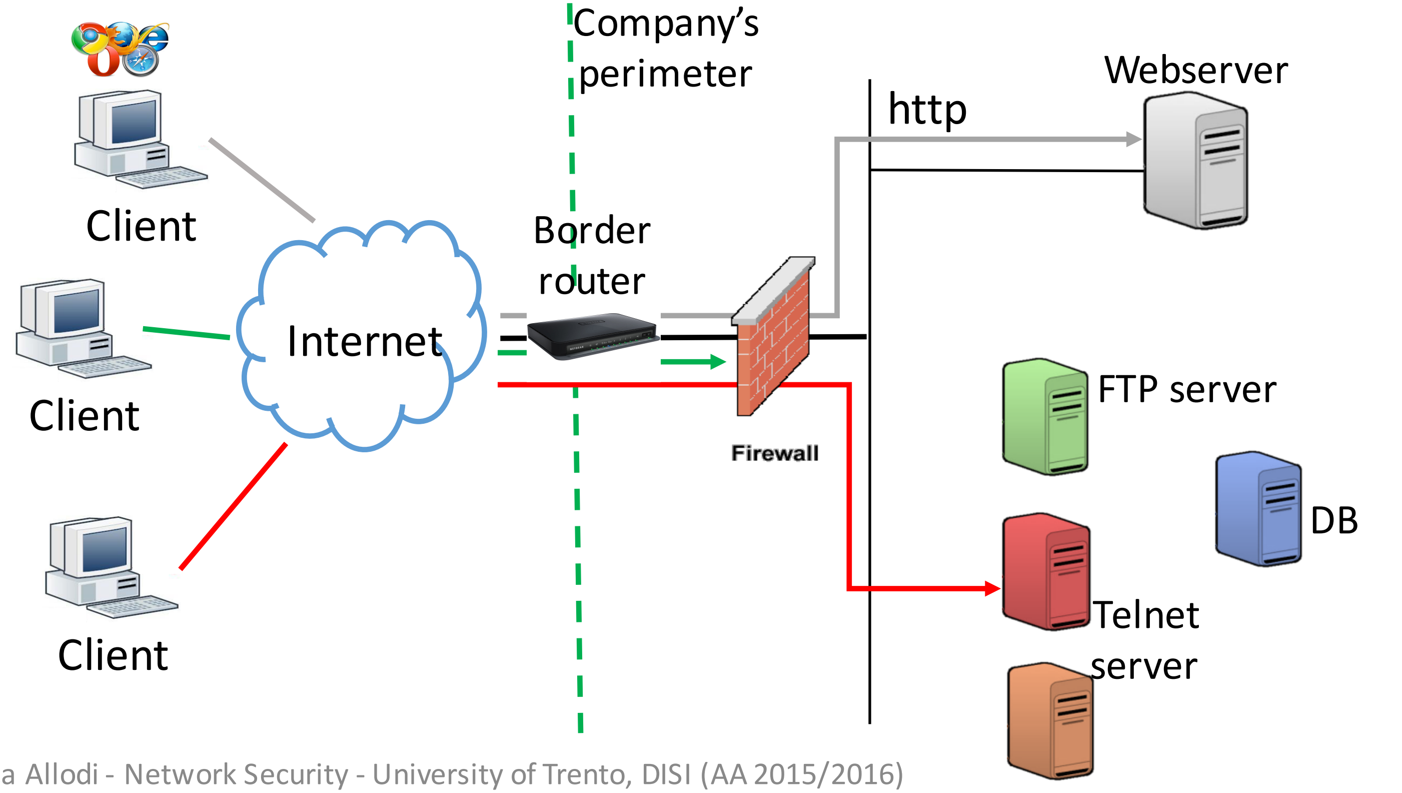
Default Policies

- Default deny:
  - *All what is not explicitly allowed is denied*
- Default permit:
  - *All what is not explicitly denied is allowed*

Default Permit

- Blacklist policy → list what is blocked
- Rules to remove/reduce services are specified when a problem is discovered
- Users have more freedom on what they can do
- Suitable for open organizations like universities or home systems

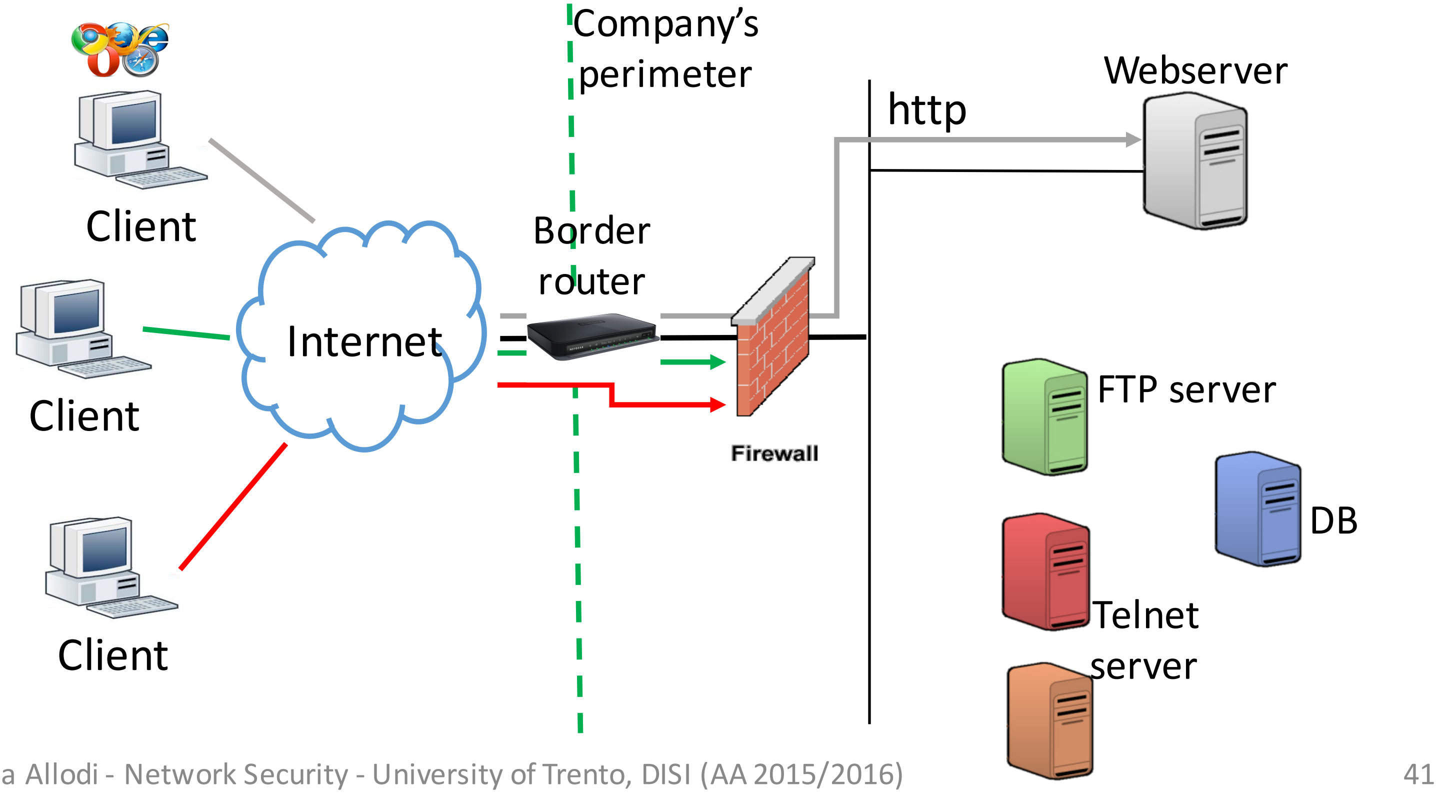
- Example permit policy
  Deny incoming ftp traffic
  Allow all



Default Permit

- Blacklist policy → list what is blocked
- Rules to remove/reduce services are specified when a problem is discovered
- Users have more freedom on what they can do
- Suitable for open organizations like universities or home systems

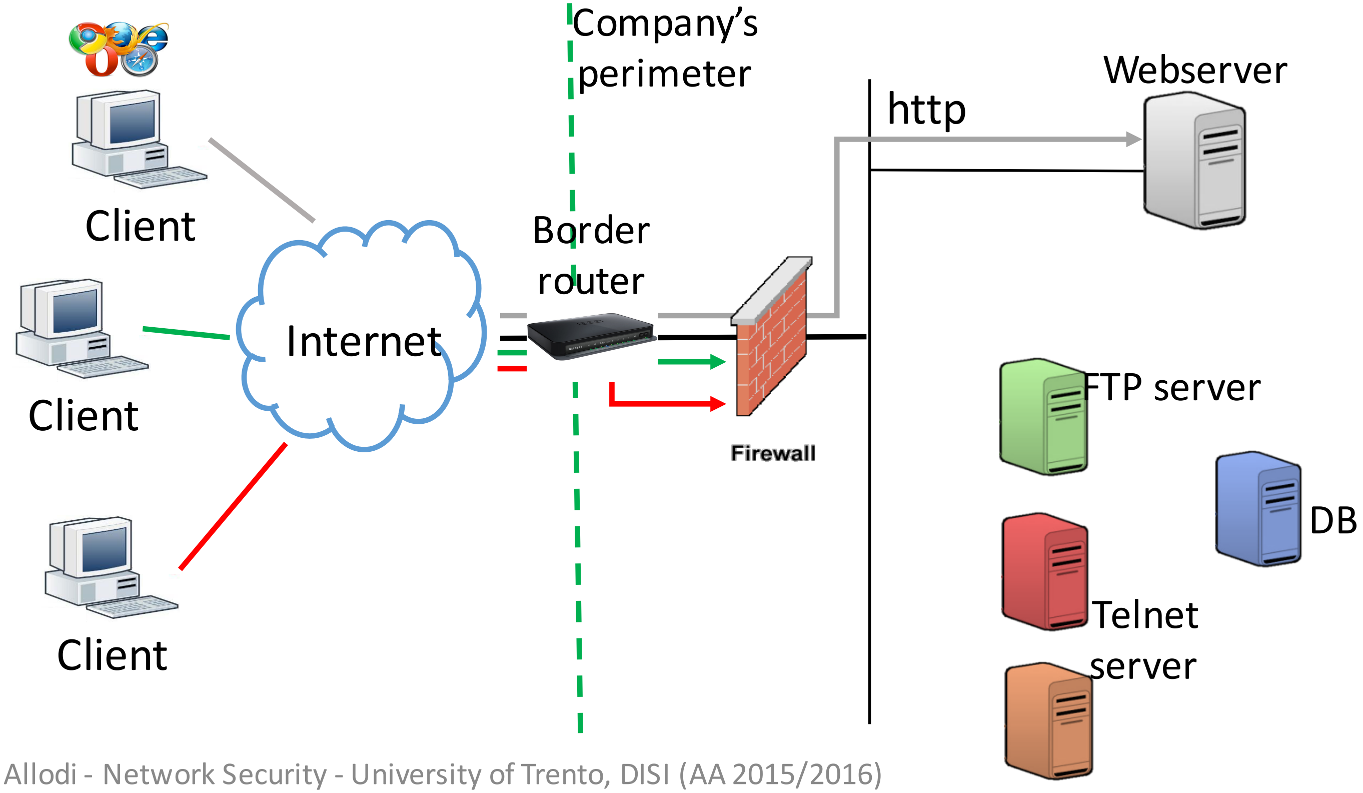
- Example permit policy
  Deny incoming ftp traffic
  Deny incoming telnet traffic
  Allow all



Default Deny

- Whitelist policy → list what is allowed
- Rules to allow a service are added after a careful analysis
- More visible to users (users are restricted at what they can do)
- Preferred default policy for business and governmental organizations

- Example deny policy
  Allow incoming http
  Deny all



Firewall Types

- Static packet filtering
- Stateful packet filtering
- Proxies
  - Application-level gateways
  - Circuit-level gateways

Static Packet Filtering

- Applies a set of rules to each *incoming IP packet* to decide whether it should be forwarded or discarded.
- *Header information* is used for filtering (e.g, protocol number, source and destination IP, source and destination port numbers, etc.)
- *Stateless*: each IP packet is examined isolated from what has happened in the past.
- Often *implemented* by a router
- Simple and fast → low demand on resources

Access lists

- Defined by CISCO format
  - Standard ACLs

    access-list $number $action $src [wild card]

    - Number → identifies rule
    - Action → accept/deny
    - Src → source ip
    - Wild card → inverse of subnet mask → says which part of the IP should be checked for and which ignored
      - e.g. 192.168.3.1 [0.0.255.255] → "0.0.3.1" is the subnet of interest
  - Extended ACLs

    access-list $number $action $type $src [wild card] $opt $dest [wild card] [log]

    - Type → IP, tcp, udp, …
    - Opt → ports for TCP/UDP, type/code for ICMP, ..
    - Log → write in log when event is triggered
- Can assign values to variables
  - e.g. internal_net:=192.168.1.0/24

http://www.cisco.com/c/en/us/support/docs/security/ios-firewall/23602-confaccesslists.html

Packet Filtering

Do we actually need this?

- Yes, if default allow
- No, if default deny

Notice that this is last in the list

- First rule that matches is used

Example of (explicit) policies:

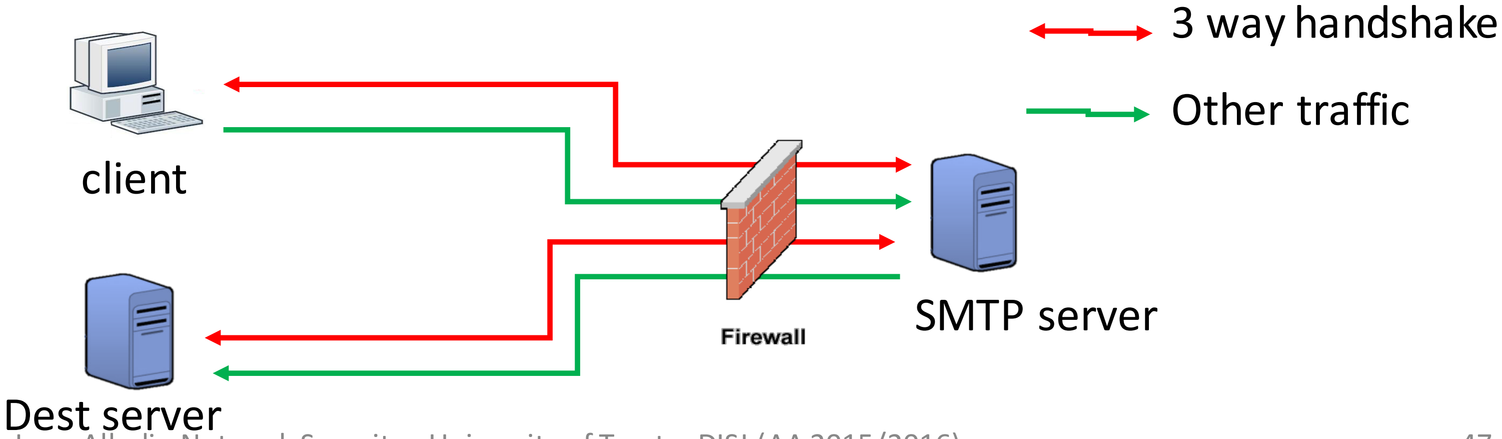
1. deny all incoming tcp connections to SSH;
2. allow outgoing TCP connections to SSH

action	src	port	dest	dport	flags	comment
allow	192.168.2.0/24	*	*	22	*	**Our outgoing traffic to remote ssh servers**
allow	*	22	192.168.2.0/24	*	S ACK	**Their SYN ACK**
allow	*	22	192.168.2.0/24	*	ACK	**Rest of communication**

action	src	port	dest	dport	flags	comment
deny	*	*	192.168.2.0/24	22	S	**We do not allow remote connections to local SSH servers**

Note of caution

- Some protocols are easy to implement
  - Clear distinction between client and server
  - Other protocols are not as straightforward
- e.g. want to restrict SMTP operations
  - SMTP server acts both as a client (receives mail) and as a server (forwards mail to next server)
  - Firewall rules must match both cases



Exercise: SMTP rules

- Explicitly allow incoming SMTP traffic from 10.1.1.1 to SMTP-srv
- Allow all outgoing SMTP traffic

action	src	port	dest	dport	flags	comment
allow	10.1.1.1	*	SMTP-srv	25	*	**allow everything from trusted client**
allow	SMTP-srv	25	10.1.1.1	*	S ACK	**allow server answer**
allow	SMTP-srv	25	10.1.1.1	*	ACK	**Allow rest of communication**
allow	SMTP-srv	*	*	25	S xor A	**Allow initiation of connection to remote SMTP**
allow	*	25	SMTP-srv	*	SA	
allow	*	25	SMTP-srv	*	A	
deny	*	*	*	*	*	

Packet Filtering: Pros and cons

- Pros
  - Transparent. It does not change the traffic flow or characteristics – either passes it through or doesn’t
  - Simple
    - Easy to implement rules to prevent IP spoofing
      - e.g. no outgoing traffic from non-private IP address space
      - Control and log attempts to remotely connect to private services
  - Cheap
- Cons
  - It does not prevent application-specific attacks
  - Unsophisticated (protects against simple attacks)
  - Calibrating rule set may be tricky
  - Limited logging