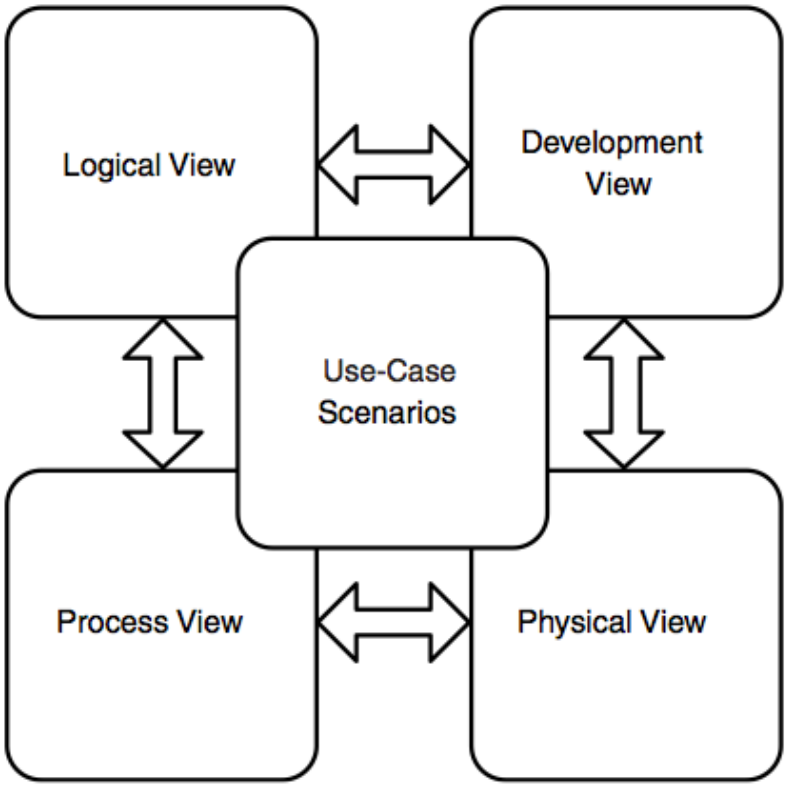
Special Topics in Security
ECE 5698

Engin Kirda
ek@ccs.neu.edu

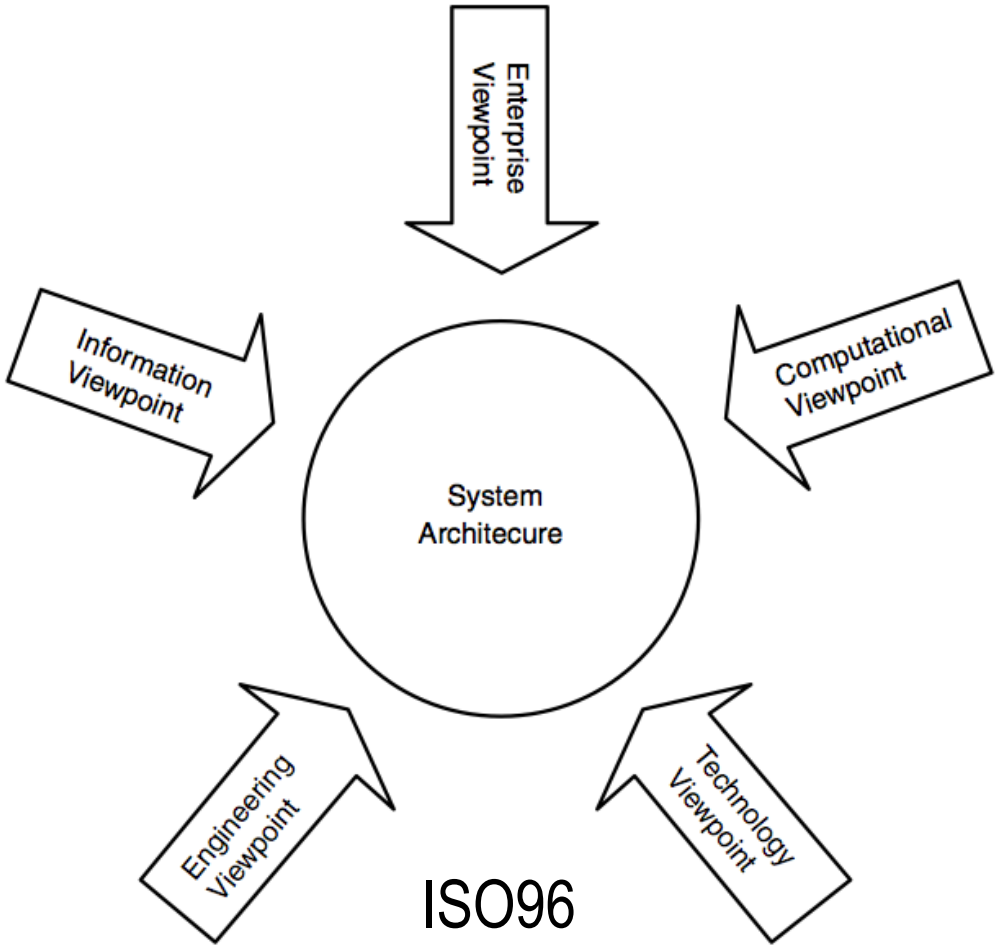
Northeastern University

Kruchten's 4+1 View Model



IEEE 1995

Reference Model for Open Distributed Processing



The Five-Level Compliance Model

- NIST Security Assessment Framework
  - Level 1: Documented policy
  - Level 2: Documented procedure
  - Level 3: Implemented procedures and controls
  - Level 4: Tested and reviewed procedures and controls
  - Level 5: Fully integrated procedures and controls
- Assessment is mostly a manual process:
  - Pre-assessment preparation
  - The assessment meeting
  - Post-assessment readout and responsibility assignments
  - Post-deployment meetings

Security Assessment Balance Sheet Model

- Provides a framework for the assessment process
  - As its name implies, it is analogous to a corporate balance sheet in an annual report
  - Designing a secure system is based on a similar balancing act between value and cost
    - Each asset has a value that is put at risk without security
    - Each security control minimizes or removes the risk of loss of value for one or more assets
    - Security costs money: 2 percent to 5 percent of the total cost of the current system release

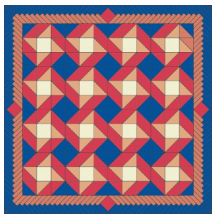
Vulnerability Identification Example

	FINDING	ASSET VALUE	PROBABILITY	RESOLUTION	CONTROL COST
1	Version of rlogin daemon on legacy system is vulnerable to buffer overflow attack.	H	L	Apply OS vendor's current security patch to the rlogin daemon.	L
2	Internet-facing Web server outside corporate firewall might be compromised.	H	H	Install corporate intrusion detection sensor on Web server. Install latest security patches. Run Tripwire on a clean document tree, and run nightly sanity checks to see whether files are compromised.	M
3	CORBA connection from application server to legacy database server is over an untrusted WAN.	H	M	Implement point-to-point IIOP over SSL connection between the two servers. Provision certificates from corporate PKI. Add performance test cases to test plan. Add certificate expiry notification as an event. Comply with corporate guidelines on cipher suites.	H
4	Administrator's Telnet session to application server might be compromised.	H	H	Require all administrators to install and use secure shell programs such as ssh and disable standard Telnet daemon.	L
5	Database allows ad hoc query access that can be compromised.	H	M	Examine application functionality to replace ad hoc query access with access to canned, stored procedures with controlled execution privileges. Parse the user query string for malicious characters.	H

[Ramachandran02]

Architectural Security Patterns

- *Patterns* are a popular concept in software engineering
- Something is a security pattern if
  - We can give it a name
  - We have observed its design repeated over and over in many security products
  - There is a benefit to defining some standard vocabulary dictated by common usage rules to describe the pattern
  - There is a good security product that exemplifies the pattern



Pattern Catalog

Pattern Catalog [Ramachandran02]

Entity	Principal						
Context holder	Session object	Cookie	Sentinel	Ticket/ Token	Role		
Service provider	Directory	Trusted Third Party	Validator				
Channel component	Wrapper	Filter	Interceptor	Proxy			
Platform component	Transport Tunnel	Distributor	Concentrator	Layer	Elevator	Sandbox	Magic

Security and Design

- Systems are often designed without security in mind
  - application programmer is often more worried about solving the problem than protecting the system
  - often, security is ignored because either the policy is generally not available, or it is easier to ignore security issues

- Organizations and individuals want their technology to survive attacks, failures and accidents
  - critical systems need to be survivable
  - This is increasing in importance
    - E.g., power plants, how secure are they?

Design Principles

- Design is a complex, creative process
- No standard technique to make design secure
- But general rules derived from experience
- 8 principles according to Saltzer and Schroeder (1975)
  "The protection of information of computer systems"

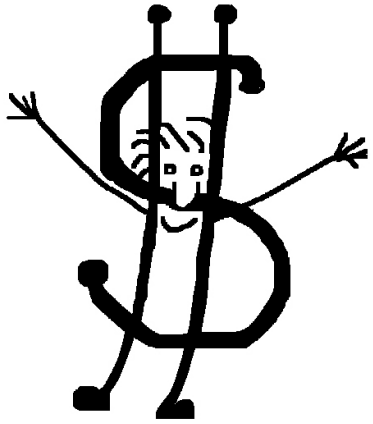
  – Economy of Mechanism
  – Fail-safe defaults
  – Complete mediation
  – Open design
  – Separation of privilege
  – Least privilege
  – Least common mechanism
  – Psychological acceptability

Economy of Mechanism

- Design should be as simple as possible
  - KISS -- keep it simple, stupid
  - Brian W. Kernighan

    "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

- Black-box / functional testing
  - treats system as a black box
  - usually does not discover security problems
  - makes white-box testing / code auditing necessary



- For successful white-box testing, simple design necessary

Fail-safe Defaults

- Allow as default action
  - grant access when not explicitly forbidden
  - in case of mistake, access allowed (often not noticed)
  - improves ease-of-use
  - wrong psychological model

- Deny as default action
  - grant access only on explicit permission
  - in case of mistake, access denied (noticed quickly)
  - improves security
  - important for firewall configurations and input validation tasks

Fail-safe Defaults

- Configuration
  - secure initial configuration
  - easy (re)configuration
- Secure initial configuration
  - no default passwords
  - no sample users
  - files are write-protected, owned by root/admin
- Error messages
  - should be very generic
  - additional information in log files

Complete Mediation

- Complete access control
  - check every access to every object
  - include all aspects (normal operation, initialization, maintenance, ..)
  - caching of checks is dangerous
  - identification of source of action (authentication) is crucial

- Trusted path
  - make sure that user is talking to authentication program
  - important for safe login (thwart fake logins)
  - Windows “control-alt-delete” sequence

Complete Mediation

- Secure interface
  - minimal
  - narrow
  - non-bypassable (e.g., check at server, not client)
- Input validation (will be very well-known soon…)
- Trust input only from trustworthy channels
  - any value that can be influenced by user cannot be trusted
    - do not authenticate based on IP source addresses / ports
    - E-mail sender can be forged (i.e., Challenge 4)
    - hidden fields or client side checks are inappropriate
    - reverse DNS lookup
  - safely load initialization (configuration)

Open Design

- Design must not be secret
  - security mechanisms must be known
  - allows review
  - establishes trust
  - unrealistic to keep mechanism secret in widely distributed systems

- Security depends on secrecy of few, small tokens
  - keys
  - passwords

- Many violations of this principle
  - especially proprietary cryptography

Separation of Privilege

- Access depends on more than one condition
  - for example, two keys are required to access a resource
  - two privileges can be (physically) distributed
  - more robust and flexible
- Classic examples
  - launch of nuclear weapons requires two people
  - bank safe
- Related principle
  - compartmentalization

Separation of Privilege

- Compartmentalization
  - break system in different, isolated parts and
  - minimize privileges in each part
  - don't implement all-or-nothing model

  → minimizes possible damage

- Sandbox
  - traditional compartmentalization technique
  - examples
    - Java sandbox (bytecode verifier, class loader, security manager)
    - virtual machines
    - paper -- Goldberg et al.,

      A secure environment for untrusted helper applications, USENIX Security Symposium,1996

Least Privilege

- Operate with least number of rights to complete task
  - minimize damage
  - minimize interactions between privileged programs
    - reduce unintentional, unwanted use
  - when misuse occurs, only few potential sources need auditing

- Minimize granted privileges
  - avoid *setuid* root programs (UNIX/Linux)
    - use groups and setgid (e.g., group *games* for high scores)
    - use special user (e.g., *nobody* for web server)
  - make file owner different from setuid user
    - taking control of process does not allow to modify program images

Least Privilege

- Minimize granted privileges
  - database restrictions
    - limit access to needed tables
    - use stored procedures

- Minimize time that privilege can be used
  - drop privileges as soon as possible
  - make sure to clear saved ID values

- Minimize time that privilege is active
  - temporarily drop privileges
  - can often be enabled by attacker as well, but protects against some kinds of attacks (e.g., file access)

Least Privilege

- Minimize modules that are granted privilege
  - optimally, only single module uses privileges and drops them
  - two separate programs
    - one can be large and untrusted
    - other is small and can perform critical operations
    - important for GUI applications that require privileges

- Limit view of system
  - limit file system view by setting new root directory
    chroot() – on Unix
  - more complete virtual machine abstraction
    BSD system call jail(2)
  - Honeypot

Least Privilege

- Do not use *setuid* scripts
  - "race condition" problems
  - Linux drops *setuid* settings
- Minimize accessible data
  - CGI scripts
    - place data used by script outside document root
    - Examples of problems…
- Minimize available resources
  - quotas
- Paper -- Provos et al., *Preventing Privilege Escalation, 12th USENIX Security Symposium*, 2003

Least Common Mechanisms

- Minimize shared mechanisms
  - reduce potentially dangerous information flow
  - reduce possible interactions

- Problems
  - beware of “race conditions”
  - avoid temporary files in global directories

Psychological Acceptability

- Easy-to-use human interface
  - easy to apply security mechanisms routinely
  - easy to apply security mechanisms correctly
  - interface has to support mental model
    - do what is expected intuitively (e.g., pers
      firewalls)
- Authentication
  - passwords
    - enforce minimum length (what is the minimum length?)
    - enforce frequent changes
  - PKI (public key infrastructure)
    - overhead vs. security

One more Design Principle

- *Separate data and control*
  - failed separation is reason for many security vulnerabilities
    - from buffer overflows to macro viruses
  - distinction between control information and data has to be clear

- Problematic
  - with automatically executing code in data files
    - JavaScript in web pages
    - automatic preview of web pages in emails
    - macros in Word
  - when using mobile code
    - code that is downloaded and executed locally

Retrofitting Applications

- Applying security techniques to existing applications
  - element of overall system design
  - when no source code available or
  - complete redesign too complicated

- Wrappers
  - move original application to new location and
  - replace it with small program or script that
    - checks (and perhaps sanitizes) command-line parameters,
    - prepares a restricted runtime, and
    - invokes the target application from its new location
  - can provide logging
  - can provide possibility for prologue and epilogue code

Retrofitting Applications

- Example wrappers
  - AusCERT Overflow Wrapper
    - exits when any command line argument exceeds a certain length
  - TCP Wrappers
    - replaces inetd (for telnet, ftp, finger, …)
    - access control
    - logging

  - sendmail restricted shell (smrsh, replacement for `/bin/sh`)
    - sendmail known for security problems
    - smrsh restricts accessible binaries
    - interestingly, vulnerable to two exploits that allow arbitrary code execution

Retrofitting Applications

- Interposition
  - insert program that we control between two pieces of software that we do not control
  - filtering of data
    - add security checks and constraints
  - network proxy
    - application policy enforcement
    - SYN flood protection
  - input sanitization (you will understand this well soon!)

Designing for Survivability

- Survivability
  - capability of a system to fulfill its mission in time despite attacks, failures, accidents
  - CERT principles

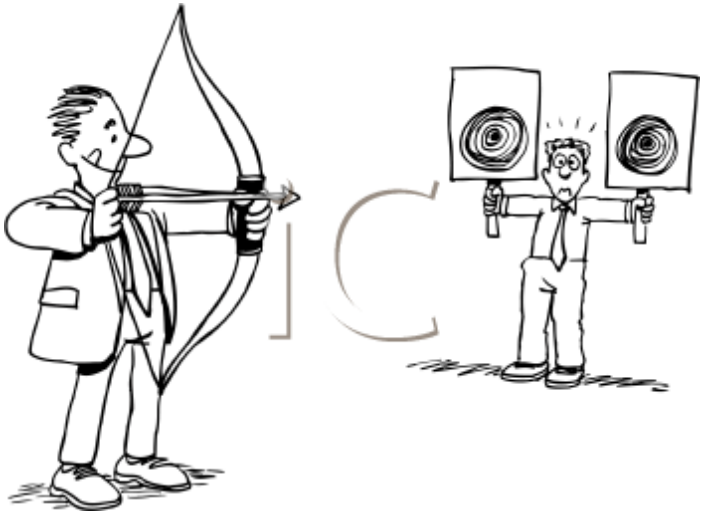
- Survivability is an enterprise-wide concern
  - at all levels of an organization
- Everything is data
  - helps understand and manage what needs to be protected
- Not all data is of equal value
  - risk assessment and focus on key points

Designing for Survivability

- Identification of users, computer systems, network infrastructure components is critical
  - secure access is granted based upon user identification
  - no matter how strong the information access control is, it is useless if based on weak identification
- Challenge assumptions to understand risks
  - “think like an intruder”
- Security knowledge in practice provides a structured approach
  - have prepared security guidelines that apply to your work in practice
- Communication skill is critical to reach all constituencies
  - most players require their own vocabulary to understand issues

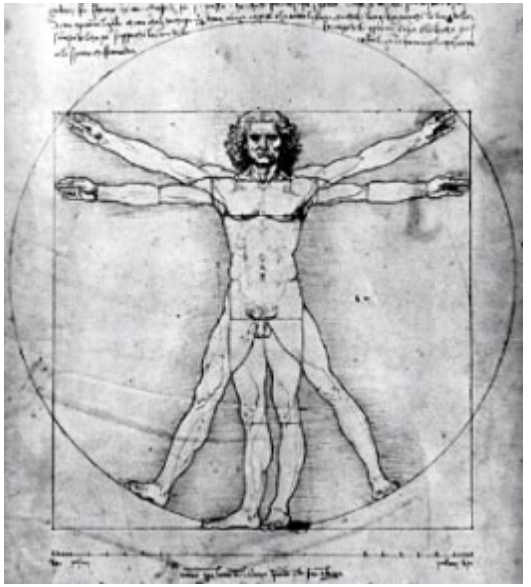
Bad Practice

- Being too specific too soon
  - without having a design, solve technical problems and start implementation
- Focus only on functionality
  - security must be built in from the beginning
- Not considering economic factors
  - ignoring the cost of security features



Bad Practice

- Not considering the human factor
  - propose solutions that users strongly dislike
    - biometric scanners instead of passwords
  - propose solutions that are annoying
    - change passwords to frequently
    - terminate idle sessions too fast
  - propose solutions that require considerable additional effort
    - producing too many alerts (e.g., snort -- “useless”)
    - require checking of many different log-files



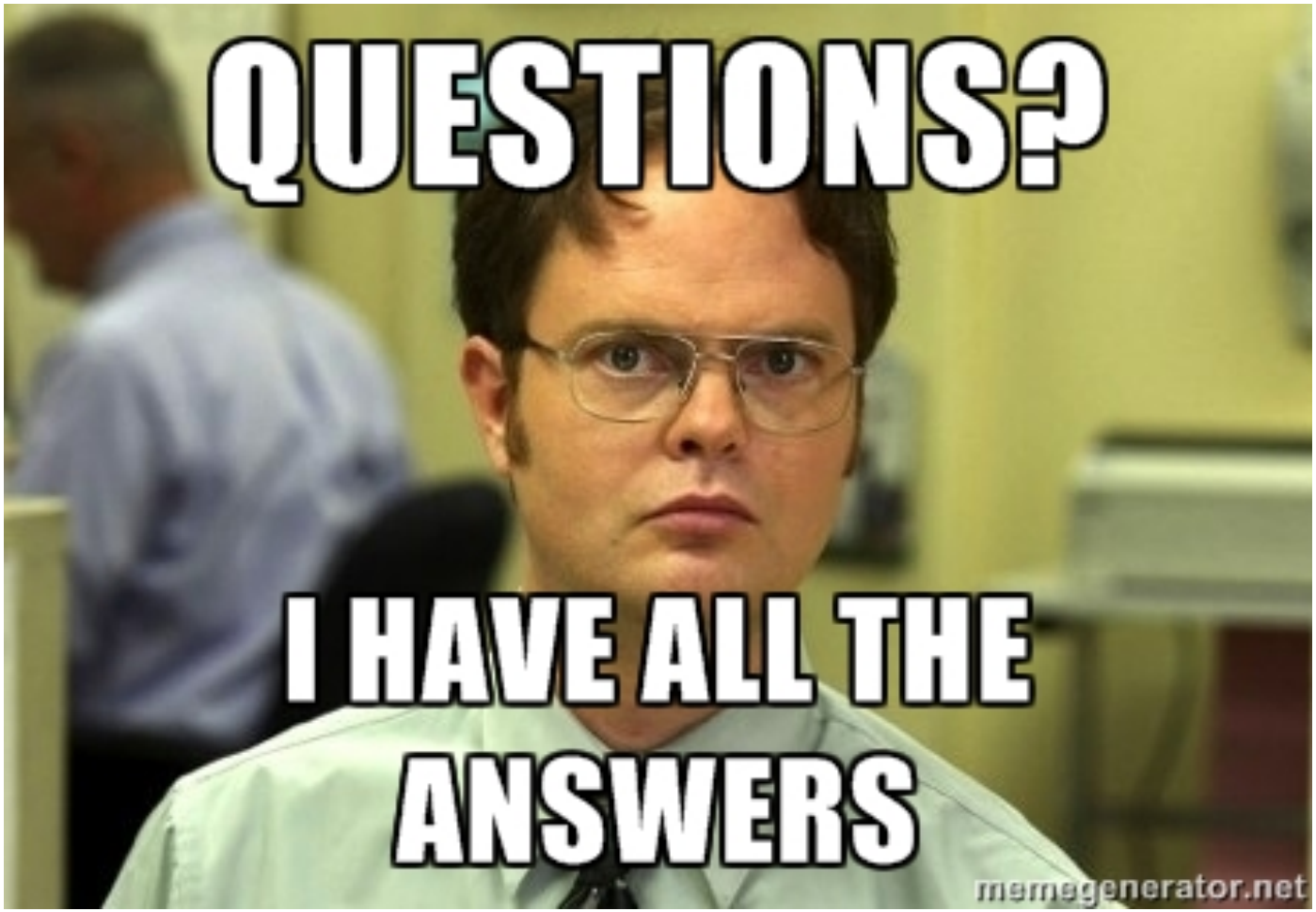
Key Messages

- Security must be considered from the start
  - security architecture
  - keeps costs to fix problems low
- Security design and architecture
  - process cannot be easily automated
  - ask important questions such as
    - what to protect (assets)
    - what to protect against (threat model)
    - how to protect (architecture)
- Most important design principles
  - least privilege
  - separate data and control

Further Reading

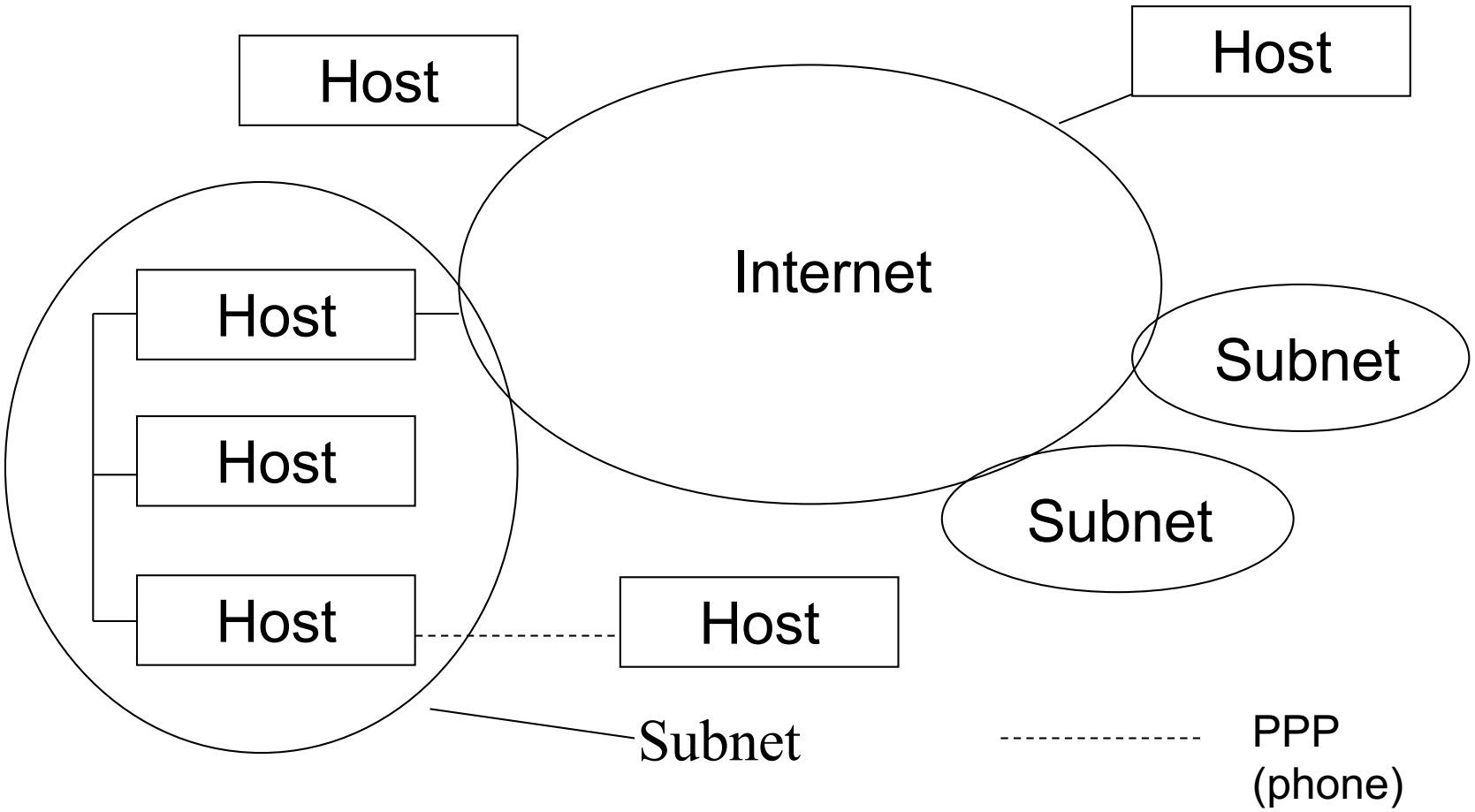
- … for the interested
- Architecture, design considerations
  - [Ramachandran02] Jay Ramachandran, Designing Security Architecture Solutions, Wiley, 2002

Questions?



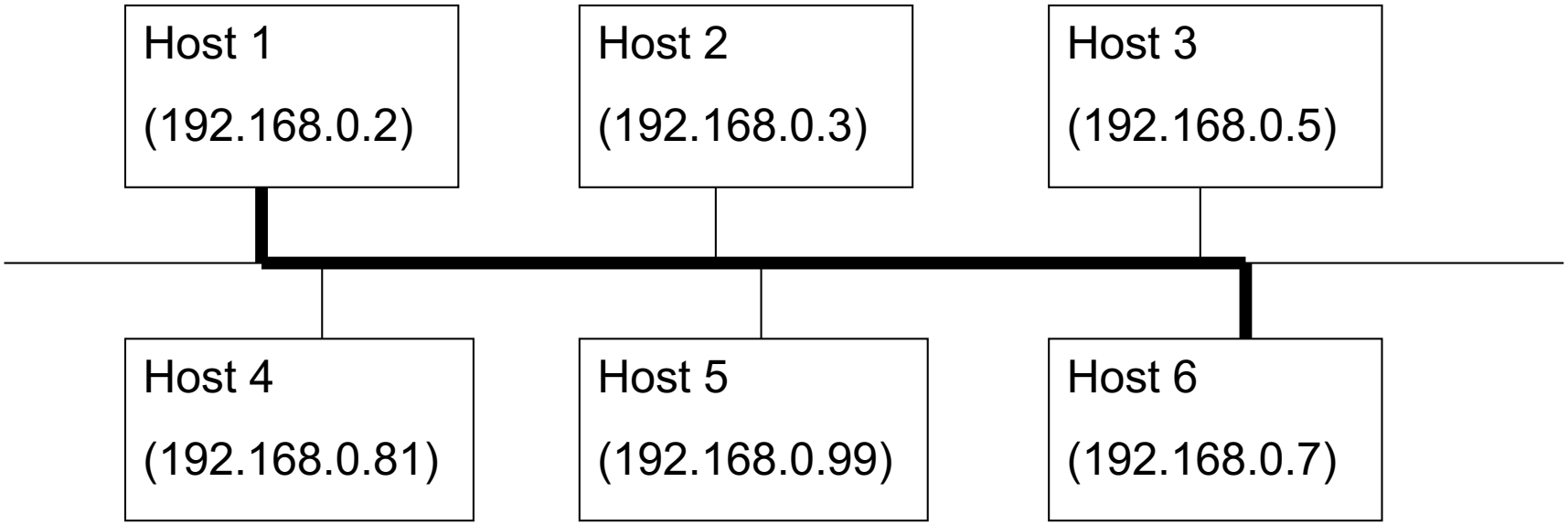
TCP IP Basics – Prep for Challenge 1

The Internet

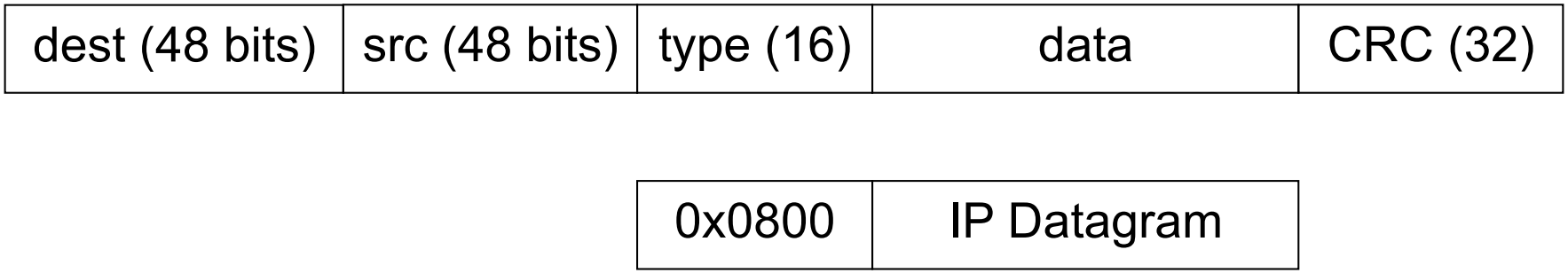


Direct IP delivery

- If two hosts are in the same physical network the IP datagram is encapsulated and delivered directly



Ethernet


dest (48 bits)
src (48 bits)
type (16)
data
CRC (32)
0x0800
IP Datagram

Ethernet

- Widely used link layer protocol
- Carrier Sense, Multiple Access, Collision Detection
- Addresses: 48 bits (e.g. 00:38:af:23:34:0f), mostly
  - hardwired by the manufacturer
- Type (2 bytes): specifies encapsulated protocol
  - IP, ARP, RARP
- Data:
  - min. 46 bytes payload (padding may be needed), max 1500 bytes
- CRC (4 bytes)

Direct IP delivery

Problem:

- Ethernet uses 48 bit addresses
- IP uses 32 bit addresses
- we want to send an IP datagram
- but we only can use the Link Layer to do this

ARP

ARP (Address Resolution Protocol)

- Service at the link-level, RFC 826
- maps network-addresses to link-level addresses
- Host A wants to know the hardware address associated with IP address of host B
- A broadcasts ARP message on physical link
  - including its own mapping
- B answers A with ARP answer message
- Mappings are cached: arp -a shows mapping

RARP

RARP (Reverse Address Resolution Protocol)

- maps link-level addresses to network-addresses
- for diskless stations to obtain their own IP address
- Service at the link-level, RFC 903

Host A wants to know its IP address (which is IP_A)

- A broadcasts RARP message on physical link
- RARP server answers with RARP answer
  - containing IP_A