DSci523: Computer Systems Assurance

Course Introduction

http://ccss.usc.edu/523

Prof. Clifford Neuman

Lecture 1
28 Aug 2020
online

Course Identification

- DSci 523
  - Assurance in Cyberspace
  - 4.0 units
- Class meeting schedule
  - 1:00-4:20pm Friday
  - Room: Online
  - Class communication
  - For now, bcn@isi.edu, include DSci523: in subject

General Course Information

- Professor office hours
  - Tuesday 2:30PM to 4:00PM
  - Other times by appointment
    Zoom link will be sent to students
  - E-mail: bcn@isi.edu

- TA/Grader for the class
  - TBA

Guidelines for Students

- Student deliverables
  - Homework assignments
  - Spontaneous class participation
  - Individual or Group Case Study on Assurance
  - Midterm exam
  - Final exam
- Read the assigned readings before class!
  - Responsible for content of assigned reading

Guidelines for Students

- All assignments are to be submitted individually
  - Help each other understand concepts
  - Help each other understand assignments
  - Work should reflect your own efforts
- Academic integrity is taken very seriously

Grading Schema

Final: 20%
Mid-Term: 20%
Class Participation: 10%
Homework Assignments: 25%
Case Study: 25%

Total 100%

Letter Grade Assignment

- A letter grade will be assigned for each assignment, project, or exam. The individual assignment scores are based on overall class performance.
- Course grade is determined by weighted calculation from the component grades.

Assurance Course Context

- Required course for Masters of Cyber Security
- Builds on hard science from previous courses
  - Includes computability, Turing machines
- Significance evident in cyber security problem
  - DSci 519: Foundations of Information Security
- Basis for the notion of trusted system design
  - DSci 525: Trusted System Design, Analysis and Development

Case Studies

- In past years the basic material for assurance was covered in INF523, and a second class INF527 applied that material in analyzing existing systems, some covered by instructor, and some by students.
- In 2016 the case studies (existing systems) covered by the instructor was integrated in-line during the teaching of the basic concepts.
- In 2017 we added back the student presented case studies… Students will choose a system or class of systems that they will use to assess the level of assurance provided, and how the architecture could be improved to increase assurance.

Questions on Course Structure

Initial Reading Assignment
(read before September 6th lecture)

- Bishop book Chapter 18, "Introduction to Assurance"
  - Computer Security Art and Science: Bishop, Matt, 2003

- Introduction to the Secure Software Development Lifecycle
  http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/

More Reading
(before September 4th)

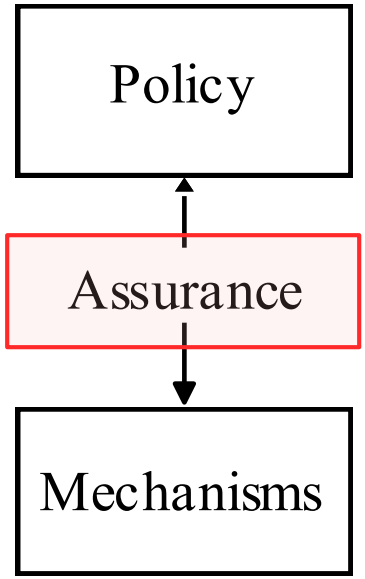
- ISACA - How Can Security Be Measured? https://www.isaca.org/Journal/archives/2013/Volume-5/Pages/How-to-Measure-Security-From-a-Governance-Perspective.aspx

- ISACA - Performing a Security Risk Assessment
- (https://www.isaca.org/Journal/archives/2010/Volume-1/Pages/Performing-a-Security-Risk-Assessment1.aspx)

Trust

- *Trustworthy* entity has sufficient credible evidence leading one to believe that the system will meet a set of requirements
- *Trust* is a measure of one's belief in trustworthiness relying on the evidence
  - To trust makes one vulnerable to violations trust.
- *Assurance* is process of building confidence that an entity meets its security requirements based on evidence provided by applying assurance techniques
  - "Meets security requirements" == Enforces policy

What is Assurance?



Statement of requirements that explicitly defines the security expectations of the mechanism(s)

Provides justification that the mechanism meets policy through assurance evidence and approvals based on evidence

Executable entities that are designed and implemented to meet the requirements of the policy

Problem Sources

1. Policy flaws
2. Requirements definitions, omissions, and mistakes
3. System design flaws
4. Hardware implementation flaws, such as wiring and chip flaws
5. Software implementation errors, program bugs, and compiler bugs
6. System use and operation errors and inadvertent mistakes
7. Willful system misuse
8. Hardware, communication, or other equipment malfunction
9. Environmental problems, natural causes, and acts of God
10. Evolution, maintenance, faulty upgrades, and decommissions

System Development Lifecycle

- Sequential stages of development and use
- Many variant SDL definitions. Here is one:
  1. Requirements gathering/definition
  2. Design
  3. Implementation (coding)
  4. Testing
  5. Release
  6. Operation
  7. Disposal

Assurance as a (non-)Priority

- Industry emphasis:
  - time to market
  - features
- Security is considered to be fixable later
  - Another way of saying that: “patch and pray”
- What is the result?

Result of Shipping Low Assurance Products

HACKERS CAN TURN YOUR HOME COMPUTER INTO A BOMB

. . . & blow your family to smithereens!

By RANDY JEFFRIES / Weekly World News

WASHINGTON — Right now, computer hackers have the ability to turn your home computer into a bomb and blow you to Kingdom Come — and they can do it anonymously from thousands of miles away!

Experts say the recent "break-ins" that paralyzed the Amazon.com, Buy.com and eBAY websites are tame compared to what will happen in the near future.

Computer expert Arnold Yabenson, president of the Washington-based consumer group National CyberCrime Prevention Foundation (NCPF), says that as far as computer crime is concerned, we've only seen the tip of the iceberg.

"The criminals who knocked out those three major online businesses are the least of our worries," Yabenson told *Weekly World News*.

"There are brilliant but unscrupulous hackers out there who have developed technologies that the average person can't even dream of. Even people who are familiar with how computers work have trouble getting their minds around the terrible things that can be done.

"It is already possible for an assassin to send someone an e-mail with an innocent-looking attachment connected to it. When the receiver downloads the attachment, the electrical current and molecular structure of the central processor unit is altered, causing it to blast apart like a large hand grenade.

"As shocking as this is, it shouldn't surprise anyone. It's just the next step in an ever-escalating progression of horrors conceived and instituted by hackers."

Yabenson points out that these dangerous sociopaths have already:

- Vandalized FBI and U. S. Army websites.
- Broken into Chinese military networks.
- Come within two digits of cracking an 87-digit Russian security code that would have sent deadly missiles hurtling toward five of America's major cities.

"As dangerous as this technology is right now, it's going to get much scarier," Yabenson said.

"Soon it will be sold to terrorists cults and fanatical religious-fringe groups.

"Instead of blowing up a single plane, these groups will be able to patch into the central computer of a large airline and blow up hundreds of planes at once.

"And worse, this e-mail bomb program will eventually find its way into the hands of anyone who wants it.

"That means anyone who has a quarrel with you, holds a grudge against you or just plain doesn't like your looks, can kill you and never be found out."

Sickos can wreak death and destruction from thousands of miles away!

Arnold Yabenson.

KABOOM! It might not look like it, but an innocent home computer like this one can be turned into a deadly weapon.

Examples

- Challenger explosion
  - Sensors removed from booster rockets to meet accelerated launch schedule
- Deaths from faulty radiation therapy system
  - Hardware safety interlock removed
  - Flaws in software design
- Bell V22 Osprey crashes
  - Failure to correct for malfunctioning components; two faulty ones could outvote a third
- Intel 486 chip
  - Bug in trigonometric functions
  - Many recent chip failures where performance was the priority – see Spectre

Result of Shipping Low Assurance Products

- Uncountable system vulnerabilities
- Endless patching
- Expensive and (nearly) useless security add-ons
- Focus on fixing the wrong things
- Continual losses to individuals, business, and government

When is High Assurance Warranted?

- Industry emphasis **made** sense for most uses
  - Consider, e.g., what most Windows systems used for
  - Another example: credit cards without pin and chip
    - "Good enough" based on estimated risk
    - (but assumptions change)
- Assurance is expensive
  - Extra time, trained staff, tools
  - May reduce product features
- High assurance required for
  - Protection of human life
  - Highly sensitive information
  - Whenever high value of potential loss

What is Assurance?

- Assurance is the generation of confidence that system satisfies (strongly and reliably enforces) security policy
- Confidence gained as result of **evidence**

Question

- A vendor advertises that its system was connected to the Internet for three months and no one was able to break into it. The vendor claims that this means the system cannot be broken into.
  - Do you share the vendor's confidence?
  - Why or why not?

What is Assurance?

- Assurance is the generation of confidence that system satisfies (strongly and reliably enforces) security policy
- Confidence gained as result of **evidence**
  - *No way to prove this with current technology*
- Must make "assurance argument"
  - Based on body of collected evidence
  - Evidence collected through assurance techniques
    - E.g., testing

System Development Lifecycle

- Sequential stages of development and use
- Many variant SDL definitions. Here is one:
  1. Requirements gathering/definition
  2. Design
  3. Implementation (coding)
  4. Testing
  5. Release
  6. Operation
  7. Disposal

Types of Assurance

- *Policy assurance* is evidence establishing security requirements in policy are complete, consistent, technically sound
- *Design assurance* is evidence establishing design sufficient to meet requirements of security policy
- *Implementation assurance* is evidence establishing implementation consistent with design

Types of Assurance (cont.)

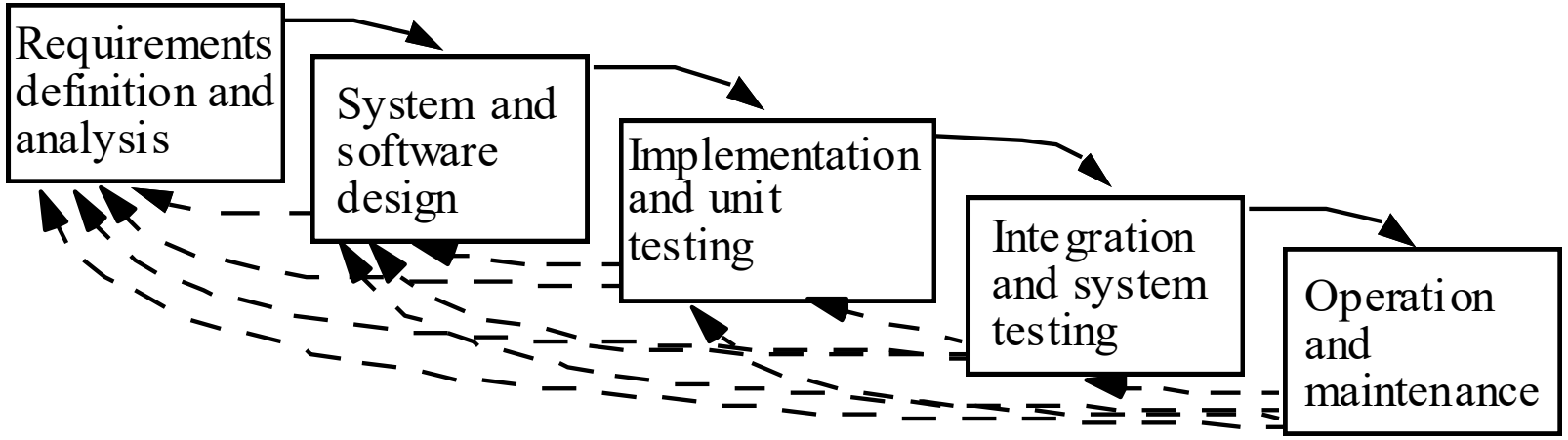
- *Operational assurance* is evidence establishing system sustains the security policy requirements during installation, configuration, and day-to-day operation
  - Also called *administrative assurance*

Discussion

- *Where and why was assurance important in the past.*
- *Where and why is assurance important NOW.*
- *What has changed*
  - *The actors*
  - *The risks*
  - *The attack surface*

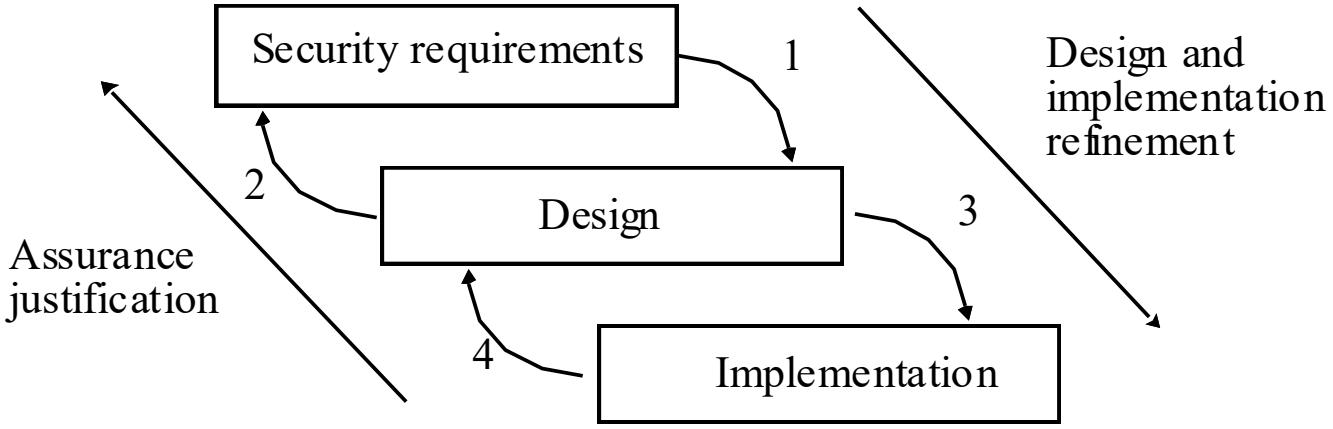
“Waterfall” Model

- Waterfall model: sequential design process
  - *With backtracking*



Assurance in SDL

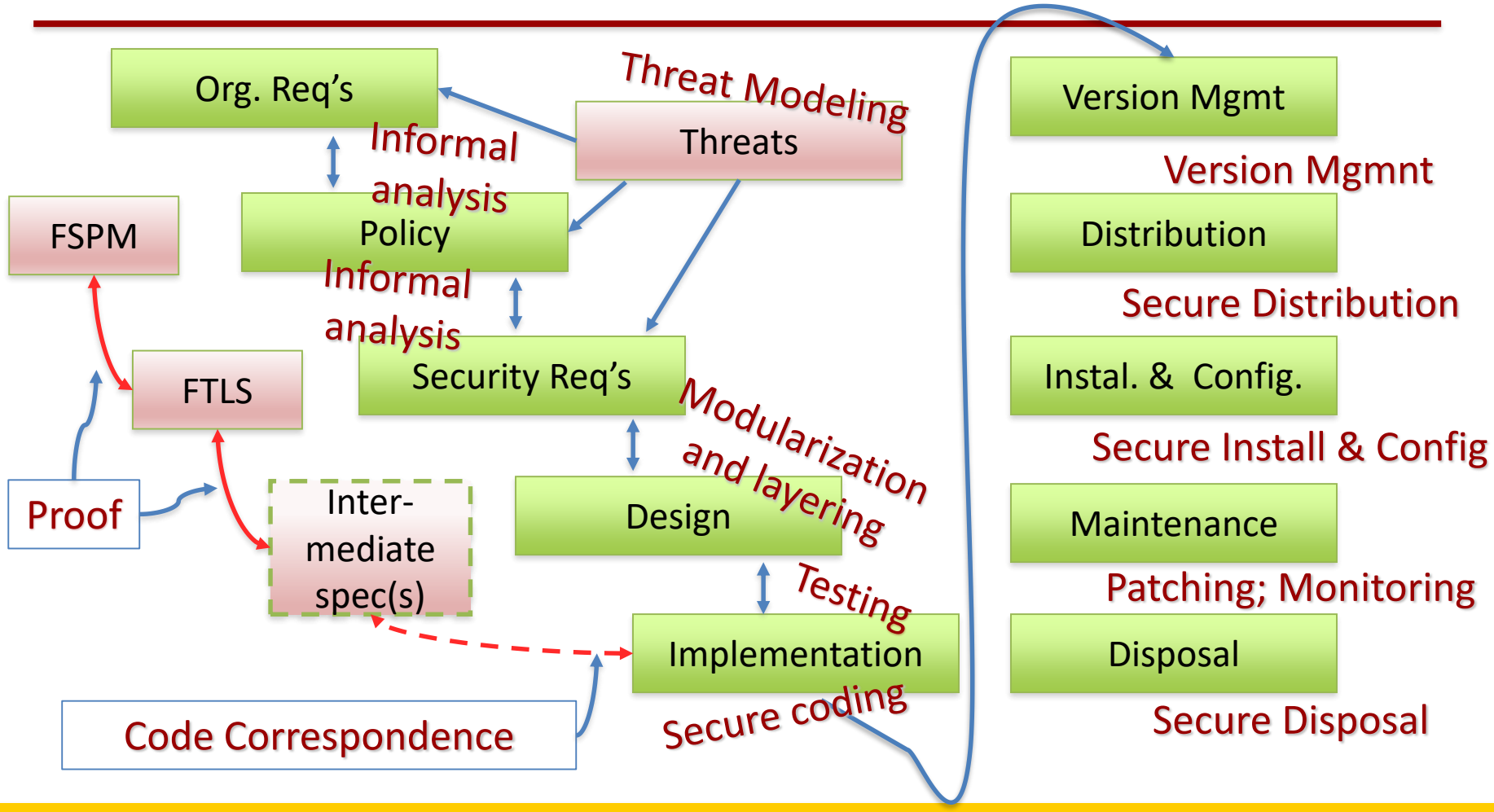
- Each stage must provide assurance justification for earlier stage
  - E.g., Does design satisfy requirements?
  - Is implementation faithful to design?
- Assurance must be built into every SDL stage



Assurance in the System Lifecycle

- *Assurance techniques* must be applied in all stages of the system lifecycle, e.g.,
  - The system's security policy is internally consistent and reflects the requirements of the organization
  - The design of the security functions is sufficient to enforce the security requirements
  - The functions are implemented correctly
  - The assurances hold up through the maintenance, installation, configuration, and other operational stages

"Assurance Waterfall"



Other Lifecycle Models

- Exploratory programming
  - Develop working system quickly
  - Used when detailed requirements specification cannot be formulated in advance, and adequacy is goal
  - No requirements or design specification, so low assurance
- Prototyping
  - Objective is to establish system requirements
  - Future iterations (after first) allow assurance techniques

Other Lifecycle Models (contd.)

- Formal transformation
  - Create formal specification
  - Translate it into program using correctness-preserving transformations
  - Very conducive to assurance methods
- System assembly from reusable components
  - Depends on whether components are trusted
  - Must assure connections and *composition*
  - Very complex, difficult to create assurance argument

Other Lifecycle Models (contd.)

- Extreme programming
  - Rapid prototyping and “best practices”
  - Project driven by business decisions
  - Requirements open until project complete
  - Programmers work in teams
  - Components tested, integrated several times a day
  - Objective is to get system into production as quickly as possible, then enhance it
  - Evidence adduced *after* development needed for assurance

Assurance Course Questions

- Strengths and weaknesses of each technique
- Which techniques give most value
- When to use specific techniques
- How to balance risk and reward

Question

- A vendor advertises that its system was connected to the Internet for three months and no one was able to break into it. The vendor claims that this means the system cannot be broken into.
- If a commercial evaluation service had monitored the testing of this system and confirmed that, despite numerous attempts, no attacker had succeeded in breaking in, would your confidence in the vendor's claim increase, decrease, or stay the same? Why? Does this constitute "proof"?

Key Points

- Assurance is critical for determining trustworthiness of systems
- Different levels of assurance, from informal evidence to rigorous mathematical evidence
- Assurance needed at all stages of system life cycle

Why Assurance in All Lifecycle Stages?

- Testing cannot (in other than trivially reduced circumstances) ever be complete
- Vendors' claims should always be suspect until sufficient evidence is provided (if it exists)
- *Assurance argument* based on totality of evidence for all stages of the system lifecycle
  - Gaps in the assurance argument are not good
  - Inconsistencies in the assurance argument are not good

Case Studies Phase 1

- Proposal Assignment for Case Study:
  - Case studies presented in two stages.
    - A 10 minute presentation on the 25th of September
      - The presentation should identify a class of systems or a particular system or product (e.g. The could, ApplePay, the i-phone, Android, SE-Linux, etc)
        - » Send me email by 9/5 for approval of topic
      - Explain (though not necessarily answer) the assurance issues that need to be met by the identified system.
      - Identify the consequences of security failure in such systems.
      - Discuss where one will look to answer those questions.
    - Topics will be grouped into related systems

Case Study Phase 2

- Students in groups will prepare a case study presentation of approximately 30 minutes per student.
  - Each student will take a section
  - Questions to be addressed include:
    - Where is the TCB(s) in the system(s)
      - How does the system utilize minimization
    - Where should the TCB(s) be.
    - What assurance techniques are applied
    - How to improve the security of the system
- These will be presented near the end of the semester.

Example Case Studies

- You can find examples of case studies by students from previous years (posted with their permission):
  - System Security Enhancements in Windows 10, Summer 2017 Case Study, Samuel Aspiranti
  - Security Enhanced LINUX, Fall 2015 Case Study - John Bush and Heather Romero

INF523: Computer Systems Assurance

Measuring Security

http://ccss.usc.edu/523

Prof. Clifford Neuman

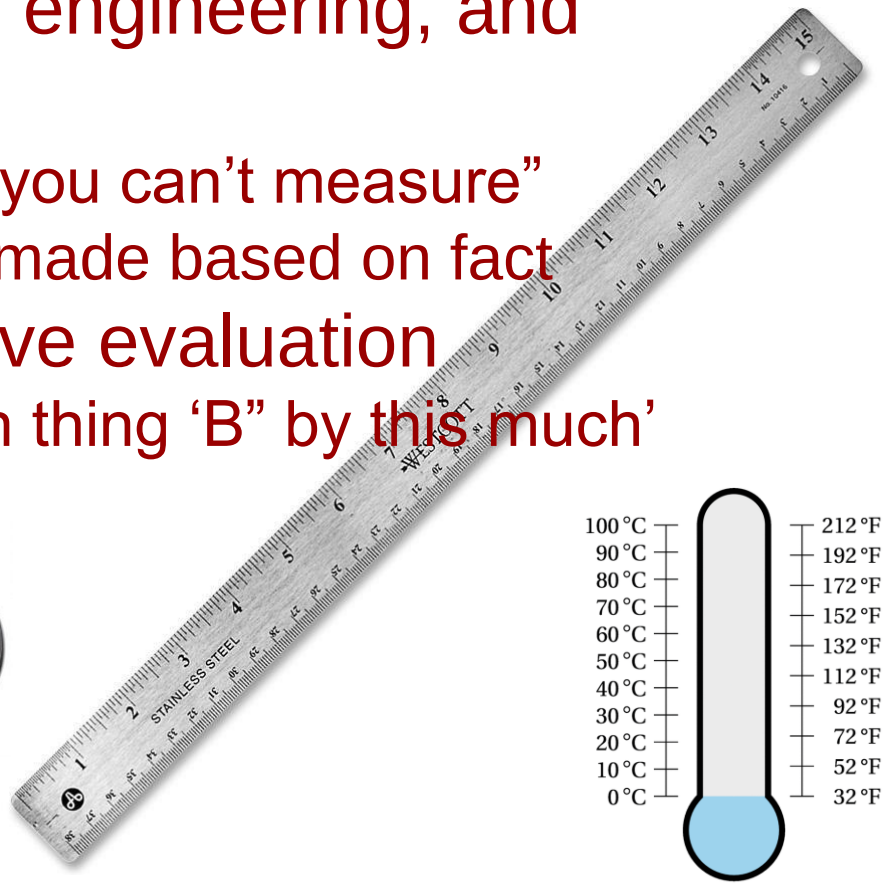
Lecture 2
4 September 2020
Online

Measurement and Assessment

- Measuring Security
  - Security metrics and their applicability to assurance
- Security Risk Assessment
  - Rationale for, and methods of, identifying and quantifying risks to an organization's information assets in order to allocate resources to increase assurance

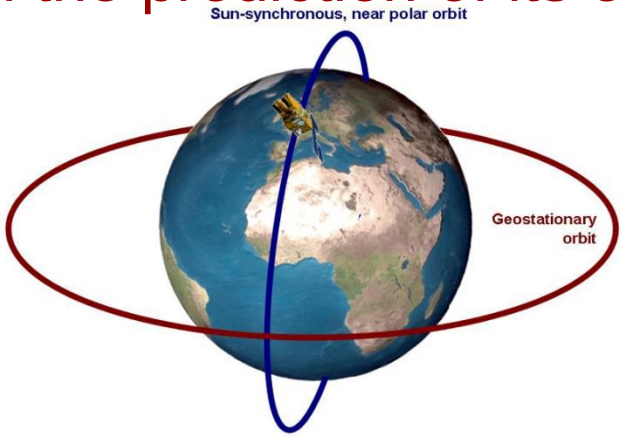
Why Measure Things?

- Cornerstone of science, engineering, and management
  - "You can't manage what you can't measure"
  - To ensure decisions are made based on fact
- Necessary for quantitative evaluation
  - "Thing 'A' is different than thing 'B" by this much'



Why Measure Things?

- The most important of science's characteristics: a record of improvement in predictive range and accuracy
  - For example, when we put a satellite in orbit, we have the scientific knowledge that guarantees accuracy and precision in the prediction of its orbit.



Measuring Computer Security

- But the record of the discipline of computer security in providing any similar level of certainty about outcomes is not good
  - **Partial Exception**: Products evaluated under Class A1 of the TCSEC (we'll talk about that later)
- Want to believe that Computer Security is a branch of engineering, the application of scientific principles or, at least, has some basis in science
- What can be basis for this belief?

Computer Security's Job

- Minimize security risk
  while
- Maximizing "business value" (however that is determined for the specific organization)

- Both require measurement and metrics

Measurement and Metrics

- *Measurement*: Dimensions, quantity, or capacity as ascertained by comparison with a standard
  - Length in meters (standard length)
  - Time in seconds (standard time interval)
- Concrete, quantitative, measure one thing

- *Metric*: Interpretation of measurements to ascertain properties or qualities of that which is measured
  - E.g., Process effectiveness, achieving of objectives
- Qualitative, often relative to a baseline

What is Measurement of Security?

- Question: When you add antivirus protection to your laptop, how many units of security do you gain?
- Answer: ???
- Are “units of security” even possible? If not, what then?

Ångstrom	1E−6 cm
Astronomical unit	1.495E8 km
Bolt (U.S., cloth)	120 ft
Cable length	720 ft
Chain (engineer's)	100 ft
Chain (surveyor's)	0.1 furlong
Cubit	18 in
Ell	45 in
Foot (U.S.)	12 in
Foot (British)	0.4 pace
Furlong	1/8 mile
Hand	4 in
Inch (U.S.)	1/12 ft
Inch (British)	1/36 yard
League	3 miles
Light year	9.46E12 km
Mile (U.S., staute)	5280 ft
Mile (nautical)	1.853 km
Nail (British)	2.25 in
Pace (British)	30 in
Parsec	3.084E13 km
Point (printer's)	1/72 in
Rope (British)	20 ft
Span	9 in

Things We'd Like to Measure

- How much more secure is an application, system, or network after adding a particular security control?
- What's the best mix of controls that will get the most security for a given investment?
- Are we secured enough?
- Are the security controls worth the price?

Current Metrics Insufficient

- Current metrics cannot answer these questions
- Many “security metrics” are really “badness-o-meters”
  - They indicate if system security is bad, but not if it is good
- E.g., vulnerability scanners
  - Detection of known vulnerabilities tells you that your system security is bad; more means it’s worse
  - Absence of detected vulnerabilities does not necessarily mean that your system security is good

Anti-phishing Training

- Anti-phishing training for employees
- Sends phishing emails to employees
- If they click on the link, they get training
- Goal is to reduce successful phishing attacks
- Later, new phishing email sent
- Measure reduction in clicking after training
- Question: If training reduces clicking on links in phishing emails by 75%, is company 75% more secure?
- From pov of IT security, does reduce workload

"Checklist" Metrics

- Based on standards documents
- Lists of "best practices"
  - E.g., NIST 800-53, *Security and Privacy Controls for Federal Information Systems and Organizations* (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf)
  - See also NIST 800-171 which is getting a lot of attention
- Counts existence of controls, but
  - Little guidance about methods, processes, or tools
  - When is each control necessary?
  - Effectiveness or value of each control?
- Useful for measuring compliance with standards
- But how much security does each control get me?

Claimed Effectiveness Metrics

- Australian "Strategies to Mitigate Targeted Cyber Intrusions" (http://www.asd.gov.au/infosec/mitigationstrategies.htm)
  - "*At least 85%* of the targeted cyber intrusions that the Australian Signals Directorate (ASD) responds to could be prevented by following the Top 4 mitigation strategies listed in our Strategies to Mitigate Targeted Cyber Intrusions:

1. use application whitelisting to help prevent malicious software and unapproved programs from running
2. patch applications such as Java, PDF viewers, Flash, web browsers and Microsoft Office
3. patch operating system vulnerabilities
4. restrict administrative privileges to operating systems and applications based on user duties."

Effective?

- I.e., At least 85% of the detected intrusions could be prevented by the top 4 controls
  - What about 100% of the undetected intrusions?
  - To be fair, they are effective against common attacks
  - Reduce IT security effort and cost
- What do the top 4 controls all have in common?
  - What is the common threat they all try to block?
    1. Prevent malicious software from running
    2. Patch applications
    3. Patch operating system
    4. Restrict user privileges
- Preventing system subversion!

Not Measuring Security

- Metrics like these do not measure security
- They measure something that "stands-in" for security, e.g.:
  - Checklists measure compliance with "best practices"
    - Belief that good practices increase security
  - Password strength measures complexity
    - Belief more complex means more secure
- Sometimes valid
  - E.g., more complex passwords are harder to guess
- Often based on assumptions and guessing
  - E.g., Metric for difficulty of attacking a system, based on estimated time and effort

Are Real Security Metrics Possible?

- Limits to accuracy and quantifiability
- Can just say “more” or “less”, without quantifying
- Consider TCSEC
  - Class A1 is more secure than B1
  - But how much more secure? Can’t quantify.

- What is fundamental about security that makes measurement so hard?

What is Security?

- A system is only secure with respect to a computer security policy
- A computer security policy denotes what is allowed and what is not allowed with respect to people accessing information stored in the system
- So computer security is the control of access by people to information stored in a computer system in order to enforce the policy
- How can we measure “control of access”?

Does That Definition Help?

- Security isn't tangible
- Nothing physical with measurable properties
  - E.g., temperature
- Not a measurable quantity in a large system
  - Like gross national product, in economics
- Security is made up by people, like rules in chess, so meaning is entirely human-defined

Many Definitions of Security

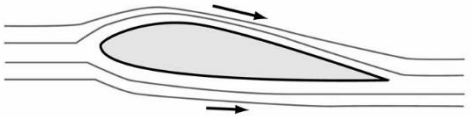
- Humans define “security” as policies
- But there are many types of policies
  - E.g., C.I.A., plus others
- Policies don’t compose into one thing
- Different components of the system have different security requirements
  - OS and applications and DNS and networking and authentication and audit and ...
  - Multi-dimensional, not additive
- Policies may contradict each other
- Keep policies separate, forget “unified theory”
  - E.g., TNI valid *assuming* secure network connections

Can't Prove Always in Secure State

- For some policies it is impossible to prove a system will always be "safe"
  - HRU result - Harrison, Ruzzo and Ullman discussed whether there is an algorithm that takes an arbitrary initial configuration and answers the following question: is there an arbitrary sequence of commands that adds a generic right into a cell of the access matrix where it has not been in the initial configuration? They showed that there is no such algorithm, thus the problem I undecidable in the general case.
- Bad choice of policy makes metric impossible
- Need to choose from classes of policies we know we can reason about
  - E.g., MAC as modeled by BLP

Need a Good Model

- Scientists use models to reason about systems
- Abstract model simplifies calculations
- Airplane wings built to satisfy fluid mechanics model
- Model doesn't work for predicting flight of wrinkled dollar bill in a windstorm
  - Wrinkled dollar bill does not fit model
- Experiments used to validate model



Need a Good Security Model

- Few organizations give precise definitions of security
  - Requirements, Policy, **Model**
- Must build information systems to meet requirements of a model
- Why do we expect to be able to predict security behavior when so many uncontrolled factors?
- Most secure systems were based on BLP model for policy
  - RM model for mechanisms

Requirements for Analyzing Systems

- Stable problems
  - E.g., ad targeting, airline scheduling
  - “Rules” are fairly well established
- Predictable range of input values
- No (or slow) drift from training data
- Large data sets

- If not stable, model needs constant updating
- Does value justify expense?

Difficult to Create Security Experiments

- Can't isolate
  - Chaotic Internet means too many confounding factors
- Too small a sample
  - Conditions widely different at different places and times
- Changes in environment happen too quickly
  - Results of an experiment no longer valid
- Can only detect and count known exploits and vulnerabilities
  - E.g., to measure effectiveness of AV
  - Can't measure what you can't see

Can't Prove a Negative

- ***Absence of evidence is not evidence of absence***
- Can only detect known exploits, so have no idea what is being missed
- Know there are uncountable vulnerabilities
- Know there are uncountable zero-days
- How can we know if we are already p0wned?
- How can we say that we are not?

- Ideally, design and build for high-assurance
  - Protect against all attacks, known and unknown

Feedback Loop

- Willful, intelligent attackers
- Actions of defenders affect actions of attackers
- Actions of attackers affect actions of defenders
- Constant change so experiments only reflect one point in time with particular conditions

- Ideally, want solutions that always work
  - Independent of attackers actions

Security is Binary

- A system is secure or it is not
  - Secure means always in secure state
  - Ivory soap is not 99.44% Secure
- Reducing exploit instances insufficient
  - Ideally, want to eliminate exploit instances
    - Anti-phishing training example from before
    - Fewer fall prey
    - That reduces # detectable instances and cost
    - But still many successful exploits
- Advanced attackers only need to succeed once

IT Risk Assessment

- The process of calculating quantitatively the potential for damage or monetary cost caused by an event that affects an organizations IT assets
- Requires
  - Identifying possible events
  - Quantifying the probability that an event will occur
  - Quantifying in $ the potential damage
- Risk = frequency(event) * damage(event)
  - Risk = Annual Loss Expectancy (ALE)
  - Product of ARO and single loss expectancy (SLE)
  - ARO = Annualized Rate of Occurence

Identifying Possible Events

- Consider
  - Threats – Potential sources of incidents
  - Vulnerabilities – Weaknesses in assets
- Event happens when threat meets vulnerability
  - Tornado and flimsy data center building
  - Hacker and unpatched windows server
- Security controls primarily focus on eliminating or mitigating vulnerabilities
- We'll talk about threat modelling later in semester

Quantifying Probability of Events

- Based on estimates
  - Estimate frequency of threat
  - Estimate existence of vulnerability
  - Estimate difficulty of exploiting vulnerability
  - Estimate cost of exploiting vulnerability
- Estimates performed by “subject matter experts”
  - “SMEs”
  - Estimate based on lots of assumptions and intuition guided by experience
- Did I say “estimate”? I meant “guess”.

Example of Estimating

- Minimum password length and complexity
  - Threat: attacker will guess a password using brute-force guessing at login
  - Vulnerability: Short, simple passwords are easy to guess
- Estimates
  - Frequency of threat: Constant
  - Existence of vulnerability: 100%
  - Difficulty of exploiting vulnerability: Based on how long would it take using guesses/sec
  - Cost of exploiting vulnerability: Proportional to difficulty
  - Damage or loss: “high”

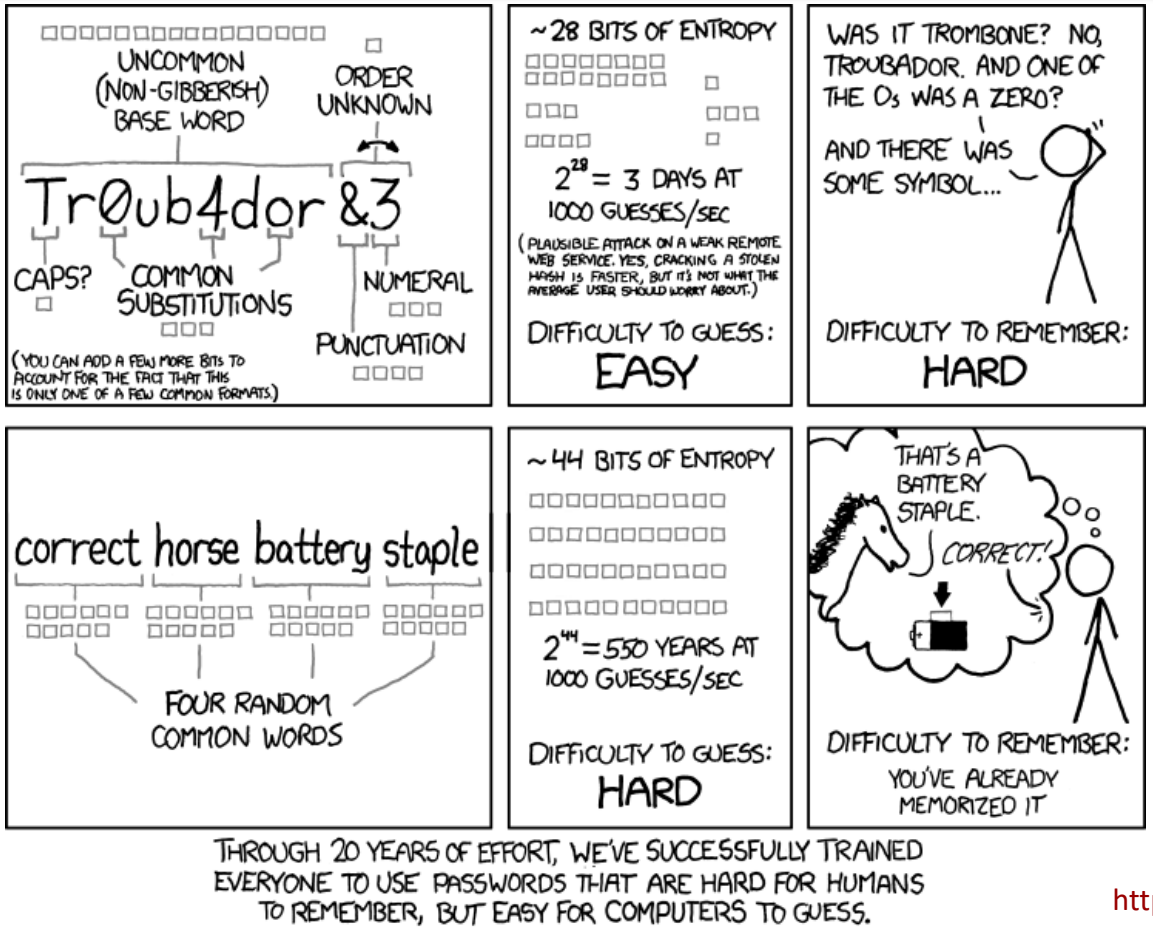
Problem with Estimates

- Frequency and vulnerability estimate does not take other controls into account
  - Limit on failed attempts
  - What if password hash stolen? A different threat!
- Potentially almost zero cost if using rainbow tables and passwords are not salted
  - If attack done by calculating hash on guessed passwords and comparing to stolen hashes
- Speed and cost of HW/SW moving target
  - Assumptions may be violated almost from the start

Mitigating Controls

- Mitigating controls or “remediations”
  - Reduce event probability or lessen impact
- New (reduced) calculated risk
- *Risk reduction* = |old_risk – new_risk|
- But remediations have costs
- *Benefit = risk reduction – cost*
- Want benefits > 0, but costs may be mandated
- Easy to miss some costs when estimating

Increasing Bits of Entropy in Passwords



https://xkcd.com/936/

Another Example

- Threat: Intruders exploiting systems over the network without detection
- Vulnerability: Can't detect attacks
- Remediation: Use IDS
- Estimates:
  - Frequency of threat: ???
  - Existence of vulnerability: 100%
  - Difficulty of exploiting vulnerability: ???
  - Cost of exploiting vulnerability: ???
  - Damage or loss: "high"

Problems with Estimates

- How to estimate frequency of attacks?
  - Published studies?
  - Do the conditions of such studies match our conditions?
- Existence of vulnerability varies according to protected resources
  - Type of system, version, patch level, configuration, …
- Difficulty/cost of exploit varies according to protected resources
- *What do SMEs do? They guess!*

IDS Mitigating Control

- Cost of IDS easy to calculate
  - Call a salesperson
- How to calculate risk reduction?
  - What percentage of total attacks does IDS see?
  - How to know? Can you trust the vendor?
  - Usually signature based, so won't see previously unknown or mutated attacks
- How to measure benefit? (Benefits are real)
- In this case, use of IDS might be mandated
- Cost must also take into account staff to monitor IDS and to review and follow-up on alerts

Measuring Security Conclusions

- No quantitative measurement possible
  - Just levels of effort to improve security
  - Effectiveness is relative, not absolute
- No way to combine all vectors into security score
- Helps to choose a good policy and model
  - Can build system secure even under changing conditions
  - Secure systems evaluated under TCSEC, based on BLP and RM, are a secure system existence proof
- Assurance is like other security measurements
  - Ultimately based on levels of effort
  - Effectiveness is relative
  - No exact measurement possible (with current technology)

Professional Orgs: SANS

- Organization that provides IT security training and certifications (https://www.sans.org/)
- Home of "Internet Storm Center" – Internet monitoring and alert system (https://isc.sans.org/)
- Lots of practical resources (http://www.sans.org/security-resources/)
- Useful newsletters (http://www.sans.org/newsletters/)

- **Assignment**: Subscribe to SANS NewsBites and @RISK newsletters

Reading for Next Week

- TCSEC, pp. 10, 50-53, 62-63, 67-68, 77-79
- Common Criteria, Part 3, pp. 15-17, 44-45
- Final Evaluation Report, Gemini Trusted Network Processor – Section 7 (GTNP-NCSC-FER-94-008.pdf)
  - [All above on DEN in "Readings" module]
- SSE-CMM/ISO 21827 Capability Maturity Model (http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html - search for "ISO 21827")
- Build Security In Maturity Model (BSIMM) (http://www.bsimm.com/)
- Microsoft Security Development Lifecycle (http://www.microsoft.com/en-us/download/details.aspx?id=29884)

Case Studies Phase 1

- Proposal Assignment for Case Study:
  - Case studies presented in two stages.
    - A 10 minute presentation on the 25th of September
      - The presentation should identify a class of systems or a particular system or product (e.g. The could, ApplePay, the i-phone, Android, SE-Linux, etc)
        - » Send me email by 9/5 for approval of topic
      - Explain (though not necessarily answer) the assurance issues that need to be met by the identified system.
      - Identify the consequences of security failure in such systems.
      - Discuss where one will look to answer those questions.
    - Topics will be grouped into related systems

Case Study Phase 2

- Students in groups will prepare a case study presentation of approximately 30 minutes per student.
  - Each student will take a section
  - Questions to be addressed include:
    - Where is the TCB(s) in the system(s)
      - How does the system utilize minimization
    - Where should the TCB(s) be.
    - What assurance techniques are applied
    - How to improve the security of the system
- These will be presented near the end of the semester.

Example Case Studies

- You can find examples of case studies by students from previous years (posted with their permission):
  - System Security Enhancements in Windows 10, Summer 2017 Case Study, Samuel Aspiranti
  - Security Enhanced LINUX, Fall 2015 Case Study - John Bush and Heather Romero

INF523: Computer Systems Assurance

Measuring Security (continued)

http://ccss.usc.edu/523

Prof. Clifford Neuman

Lecture 2
4 September 2020
Online

Foundations of Computer Security

1. A security policy that states the laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information
2. The functionality of internal mechanisms to enforce that security policy
3. Assurance that the mechanisms correctly enforce the security policy

Review of Key Concepts

- A system *security policy* is a statement of requirements that defines security expectations
- *Cyber assurance* is trust that system correctly enforces security policy
- *Trust* must be gained through evidence, using assurance techniques
- *Assurance techniques* must be applied at all stages of the system lifecycle

Assurance in the System Lifecycle

- The system's security policy is internally consistent and reflects the requirements of the organization
- There are sufficient security functions to support the security policy
- The security functions meet a desired set of properties (and only those properties)
- The functions are implemented correctly
- The assurances hold up through the manufacturing, delivery, and other stages of the system lifecycle

High vs. Low Assurance

- We provide assurance through techniques such as structured design processes, documentation, and testing
- Higher assurance through use of more, and more rigorous, processes, documentation, and testing
  - Intuitively, compare a little testing vs. a great deal of testing
- There are some fundamental ways to organize these processes to make the job easier and to have a more robust product

A Good Start

- Implement a reference monitor
  - Tamperproof
  - Always invoked
  - "**Small enough to be subject to independent testing, the completeness of which can be assured**" (TCSEC)
- *Trusted Computing Base (TCB)*
  - Combination of hardware, software, and firmware that is responsible for enforcing the system's security policy
- **Minimization of the complexity** in the TCB is a goal
- That puts more code outside the TCB, but so what?
  - **We are concerned here with the security policy only**

Question

- A vendor says they have thoroughly tested their system and found no flaws. They say their system can be “trusted” with sensitive information. Would you believe it?
  - Why or why not?

Question

- If Microsoft tomorrow was to announce that "we've identified the security flaws in Windows 10 and came up with new security requirements that we implemented in Windows 11, so Windows 11 is totally secure", would you believe it?
  - Why or why not?
  - If not, what would it take for you to believe it?

Question

- Your company wants to build a “secure” application that will run on Windows (version 10 and up)
  - You can expend as much effort, tools, and other resources necessary to make the application secure
  - You determine the security req’s, model threats, create a policy, generate the security req’s and use formal methods to ensure the security features map to the policy, create a design and use formal methods to map the design to the security req’s, implement the design and formally “prove” that the code satisfies the specification.
- Is your application high-assurance?

What do these examples tell us?

- Testing cannot (in other than trivially reduced circumstances) ever be complete
- Vendors' claims should always be suspect until sufficient evidence is provided (if it exists)
- You can't spin gold out of straw
  - Must consider entire TCB, not just components
- *Assurance argument* based on totality of evidence for all stages of the system lifecycle
  - Gaps in the assurance argument are not good
  - Inconsistencies in the assurance argument are not good

Evaluating Assurance Arguments

- How does a normal purchaser of software and systems evaluate vendor security claims and evidence?

- Depend on 3rd-party expert evaluators
  - Presumed: Trained, experienced, good judgment, unbiased

Topics Covered in this Lecture

- **Assurance Requirements in Evaluation Criteria**
  - Assurance requirements at different evaluation levels
- **Capability Maturity Models**
  - An approach for assessing the capability of a vendor to produce a secure system
- **Microsoft's SDLC**
  - A contemporary, real-world assurance process at a major software company

An Example Evaluation Criteria

- The Orange Book View of Assurance
  - Orange Book = TCSEC (http://csrc.nist.gov/publications/history/dod85.pdf)
- No longer used, but good example of what is needed to make assurance arguments

Orange Book Assurance

- The Assurance Control Objective
  - Systems that are used to process or handle classified or other sensitive information must be designed to guarantee correct and accurate interpretation of the security policy and must not distort the intent of that policy. **Assurance must be provided that correct implementation and operation of the policy exists throughout the system's life-cycle.**
    - Operational Assurance
    - System Architecture
    - System Integrity
    - Covert Channel Analysis
    - Trusted Facility Management
    - Trusted Recovery

A Complication

- Should we always expect very high assurance evidence in every instance of security solutions / technology?
  - What is the case for? What is the case against?
- Consider the padlock industry
  - There are many different locks on the market
  - There is the notion of best, better, good, good enough and sort of ok
  - We make a risk judgment every time we buy one
  - Implicit assurance argument about sufficiency to mitigate the threat and “fit for purpose” (trustworthiness)
  - But the assurance is not the same across the lock space

Subtleties – Balanced Assurance

- Suppose you have a really strong, high assurance perimeter control system. You've made a heavy investment to make sure only permitted individuals are able to have access through this perimeter. By allowing those individuals through the perimeter you have acknowledged great trust. Each of them has an office inside the perimeter. each office has a lock on it. What threat does the office lock mitigate?
- Does the lock on the office need to be a super strong high assurance lock, comparable with the perimeter?

Subtleties - Composability

- Consider a high security jail that is built by assembling modules from various different (likely by a lowest bid, but no kickback suppliers)
- There are modules for the walls (with and without windows) doors, cells, services (food, medical and other). The modules "snap" together into the jail.
- You, of course, are worried about the management of this jail and very worried that someone will try to break out of this jail.
  - Each module is constructed to your specification with your assurance requirements and validated at the manufacturer
  - What would be the assurance argument for the jail as a unit?
  - How would you go at this problem
  - How would you decide if / when the jail was fit for purpose?

Subtleties – Assurance and "the right stuff"

- Vendors market what sells, independently of what matters
- Vendors might try to pad the assurance with stuff that is not overly germane to your specific needs
  - For example paint quality on patrol cars, The vendor could make a big deal out of this but that is not one of your most important concern
- Consider intrusion detection systems. You want an IDS that stops all intrusions. What would be an assurance argument that the Vendor might present to you to substantiate that claim and what might be the important part(s) he/she might leave out?

TCSEC Classes

- D – Minimal Protection
- C – Discretionary Protection
  - C1 – Discretionary Security Protection - DAC
  - C2 – Controlled Access Protection – DAC + audit, etc.
- B – Mandatory Protection
  - B1 – Labeled Security Protection (has MAC labels)
  - B2 – Structured Protection (FSPM)
  - B3 – Security Domains (implements RM)
- A – Verified Protection
  - A1 – Verified Design (formal design, spec, and verify)

TCSEC summary Security Policy

Figure 1.: Summary of the TCSEC

SECURITY POLICY	C1	C2	B1	B2	B3	A1
Discretionary Access Control						
Object Reuse						
Labels						
Label Integrity						
Exportation of labelled information						
Exportation to Multi-Level Devices						
Exportation to Single-Level Devices						
Labelling Human-Readable Output						
Mandatory Access Control						
Subject Sensitivity Labels						
Device Labels						

LEGEND

No additional requirements	
New or enhanced requirements	
No requirements	

TCSEC summary Accountability

Figure 1.: Summary of the TCSEC

	C1	C2	B1	B2	B3	A1
ACCOUNTABILITY						
Identification and Authentication						
Audit						
Trusted Path						

LEGEND

No additional requirements	
New or enhanced requirements	
No requirements	

TCSEC summary Documentation

Figure 1.: Summary of the TCSEC

	C1	C2	B1	B2	B3	A1
DOCUMENTATION						
Security Features User's Guide						
Trusted Facility Manual						
Test Documentation						
Design Documentation						

LEGEND

No additional requirements	
New or enhanced requirements	
No requirements	

TCSEC summary Assurance

Figure 1.: Summary of the TCSEC

ASSURANCE	C1	C2	B1	B2	B3	A1
System Architecture						
System Integrity						
Security Testing						
Design Specification and Verification						
Covert Channel Analysis						
Trusted Facility Management						
Configuration Management						
Trusted Recovery						
Trusted Distribution						

LEGEND

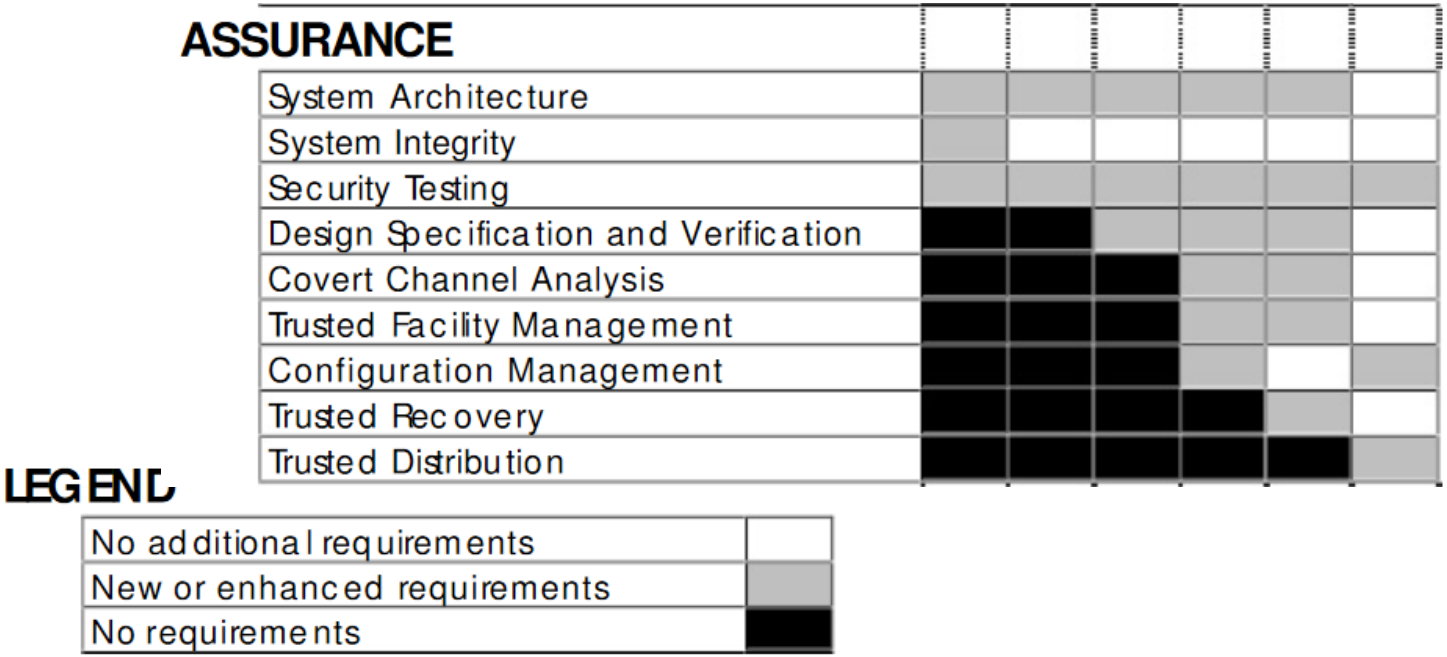
No additional requirements	
New or enhanced requirements	
No requirements	

Orange Book Assurance Requirements

- Orange book has different assurance requirements for different classes
- Except for system integrity, each of the assurance measures is graded
  - The measure is incrementally increased as the threat mitigation expectation of the system increases
- Logical grouping of the assurance measures.
  - Design Specification and Verification is not part of the DAC-only policy systems (Why?)
  - In fact, most of the assurance measures only apply to systems that provide MAC policy enforcement
  - I.e., they apply to the machine in the middle (the RM)

Example Local Grouping

- A1's chief concern is subversion, so yet more requirements for, e.g., trusted distribution



Example of Graded Assurance Measures

- We'll look at Security Testing assurance requirements
- The security testing "ramp"
  - Higher assurance for higher security class

Class C1 Security Testing

- **The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. (See the Security Testing Guidelines.)**

Class C2 Security Testing

- The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. **Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data. (See the Security Testing guidelines**.)

Class B1 Security Testing

- The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. **A team of individuals who thoroughly understand the specific implementation of the TCB shall subject its design documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the TCB to read, change, or delete data normally denied under the mandatory or discretionary security policy enforced by the TCB; as well as to assure that no subject (without authorization to do so) is able to cause the TCB to enter a state such that it is unable to respond to communications initiated by other users. All discovered flaws shall be removed or neutralized and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced. (See the Security Testing Guidelines.)**

Class B2 Security Testing

- The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. A team of individuals who thoroughly understand the specific implementation of the TCB shall subject its design documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the TCB to read, change, or delete data normally denied under the mandatory or discretionary security policy enforced by the TCB; as well as to assure that no subject (without authorization to do so) is able to cause the TCB to enter a state such that it is unable to respond to communications initiated by other users. **The TCB shall be found resistant to penetration**. All discovered flaws shall be corrected and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced. **Testing shall demonstrate that the TCB implementation is consistent with the formal top level specification.** (See the Security Testing Guidelines.)

Class B3 Security Testing

- The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. A team of individuals who thoroughly understand the specific implementation of the TCB shall subject its design documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the TCB to read, change, or delete data normally denied under the mandatory or discretionary security policy enforced by the TCB; as well as to assure that no subject (without authorization to do so) is able to cause the TCB to enter a state such that it is unable to respond to communications initiated by other users. The TCB shall be found resistant to penetration. All discovered flaws shall be corrected and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced. Testing shall demonstrate that the TCB implementation is consistent with the formal top level specification. (See the Security Testing Guidelines.) **No design flaws and no more than a few correctable implementation flaws may be found during testing and there shall be reasonable confidence that few remain.**

Class A1 Security Testing

- The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. A team of individuals who thoroughly understand the specific implementation of the TCB shall subject its design documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the TCB to read, change, or delete data normally denied under the mandatory or discretionary security policy enforced by the TCB; as well as to assure that no subject (without authorization to do so) is able to cause the TCB to enter a state such that it is unable to respond to communications initiated by other users. The TCB shall be found resistant to penetration. All discovered flaws shall be corrected and the TCB retested to demonstrate that they have been eliminated and that new flaws have not been introduced. Testing shall demonstrate that the TCB implementation is consistent with the formal top level specification. (See the Security Testing Guidelines.) No design flaws and no more than a few correctable implementation flaws may be found during testing and there shall be reasonable confidence that few remain. **Manual or other mapping of the FTLS to the source code may form a basis for penetration testing.**

Security Testing Guidelines – Division C

- At least two people with Bachelor degrees in CS
- Must be familiar with “flaw hypothesis” testing methodology (a form of pen-testing)
  - Create list of possible flaws through analysis of specs and documentation of a system
  - Prioritize based on likelihood and ease of exploit
- Must carry out system developer-defined tests
- Must independently design and implement at least 5 tests
  - 1 month <= testing <= 3 months
  - 20 hours <= testing for each team member

Security Testing Guidelines – Division B

- At least two people with Bachelor degrees in CS
- At least one person with Master's Degree in CS
- Must be fluent in TCB implementation language(s)
- Must be experienced with assembly language
- Must have completed system developer's internals course for the system
- At least one team member must have completed a security test on another system
- Team must independently design and implement at least 15 tests
  - 2 months <= testing <= 4 months
  - 30 hours <= testing for each team member

Security Testing Guidelines – Division A

- At least one person with Bachelor degree in CS
- At least two people with Master's degrees in CS
- At least one team member must be familiar enough with the system hardware to understand diagnostic programs and HW documentation
- At least 2 team members must have completed a security test on another system
- At least one team member must have demonstrated expertise on the system under test sufficient to, e.g., add a device driver
- Team must independently design and implement at least 25 tests
  - 3 months <= testing <= 6 months
  - 50 hours <= testing for each team member

Example FER for Class A1 System

- Final Evaluation Report (FER) for Gemini Trusted Network Processor
- The FER provides descriptions of the following system aspects:
  - Architecture HW (e.g., segmentation and rings) and SW
  - Design
  - Mapping to policy model (BLP)
  - Assurance
    - Architecture – modularization and layering, data hiding, minimization
    - System Integrity
    - Covert channel analysis and testing
    - Trusted Recovery
    - Security Testing – functional and exception testing
    - Design specification and verification
    - Configuration management and maintenance
    - Trusted Distribution

Assurance Takeaway from the TCSEC

- Tied to security policy and FSPM
- Based on notion of TCB
  - All the HW/SW/FW in the system that enforces the policy
  - The other stuff doesn't matter from security pov
- Identifies SDLC + assurance techniques for each step
- Trades off level of assurance against required protection
- Permits different levels of effort against different layers of the system ("balanced assurance")
- Considers composition of parts (TNI)

Topics Covered in this Lecture

- **Assurance Requirements in Evaluation Criteria**
  - Assurance requirements at different evaluation levels
- **Capability Maturity Models**
  - An approach for assessing the capability of a vendor to produce a secure system
- **Microsoft's SDLC**
  - A contemporary, real-world assurance process at a major software company

Capability Maturity Models

- How can a user assess the security of a product?
  - After lengthy, third-party evaluation (but product may be nearly obsolete by then)
  - Immediately, but assurance rests on claims by vendor
- Improve assurance and time-to-market by *pre-reviewing security engineering processes of vendor*
  - Third-party review of vendor security engineering processes (capabilities)
  - Focus on measuring organization competency (maturity) and improvements

Capability Maturity Models (2)

- Goals:
  - Continuity - knowledge acquired in previous efforts is used in future efforts
  - Repeatability - a way to ensure that projects can repeat a successful effort
  - Efficiency - a way to help both developers and evaluators work more efficiently
  - Assurance - confidence that security needs are being addressed.

Capability Maturity Models (3)

- Focus on existence of process
- Organizations appraised by third-party
- Score based on number and sophistication of practices followed
- Goal for vendor is to be appraised high for competitive advantage
- Acquirers can put required CMM level in RFPs
- Similar to “six sigma” or ISO 9000 certification for quality and process improvement

System Security Engineering - Capability Maturity Model (SSE-CMM, ISO/IEC 21827)

- Covers entire organization, including management as well as engineering
- Based on observed engineering best practices at over 50 large organizations (including multi-nationals)
- Addresses the complete product life cycle:
  - Concept definition
  - Development
  - Production
  - Utilization
  - Support
  - Retirement

System Security Engineering

Capability Maturity Model (SSE-CMM, ISO/IEC 21827)

- Two dimensions: *domain* and *capability*
- Domain is "base practices" of security engineering
  - E.g., Base Practice 05.02, "Identify System Security Vulnerabilities"
- Capability is "generic practices" that should be part of base practices
  - E.g., Generic Practice 2.1.1, "Allocate Resources"
- Intersection indicates an organization's capability to perform a particular activity
  - E.g., "Does the organization allocate resources for use in identifying system security vulnerabilities?"

Base Practices

- Apply to entire life cycle
- Represents a “best practice” in the security community
- Organized into Process Areas
  - Not all organizations have same needs or goals
  - Some provide products, some systems, others services

Process Areas for Systems Security Engineering

- PA01 Administer Security Controls
- PA02 Assess Impact
- PA03 Assess Security Risk
- PA04 Assess Threat
- PA05 Assess Vulnerability
- PA06 Build Assurance Argument
- PA07 Coordinate Security
- PA08 Monitor Security Posture
- PA09 Provide Security Input
- PA10 Specify Security Needs
- PA11 Verify and Validate Security
- Additional process areas for project and org. practices

PA05 - Assess Vulnerability

- Description:
  - Identify and characterize security vulnerabilities
    - Analyze system assets
    - Define specific vulnerabilities
    - Provide an assessment of the overall system vulnerability
  - Performed any time during a system's life-cycle
- Goals:
  - An understanding of system security vulnerabilities within a defined environment is achieved

PA05 - Assess Vulnerability

- Base Practice List:
  - BP.05.01 Select the methods, techniques, and criteria by which security system vulnerabilities in a defined environment are identified and characterized
  - BP.05.02 Identify system security vulnerabilities
  - BP.05.03 Gather data related to the properties of the vulnerabilities
  - BP.05.04 Assess the system vulnerability and aggregate vulnerabilities that result from specific vulnerabilities and combinations of specific vulnerabilities
  - BP.05.05 Monitor ongoing changes in the applicable vulnerabilities and changes to their characteristics

BP.05.02 – Identify Vulnerabilities

- Description: The methodology of attack scenarios (description of specific attacks) as developed in BP.05.01 should be followed to the extent that vulnerabilities are validated. All system vulnerabilities discovered should be recorded.
- Example Work Products:
  - Vulnerability list describes the vulnerability of the system to various attacks
  - Penetration profile includes results of the attack testing (e.g., vulnerabilities)

SSE-CMM Capability Levels

0 INITIAL

1 PERFORMED INFORMALLY

- **Base practices performed**

2 PLANNED & TRACKED

- **Planning performance**
- **Disciplined performance**
- **Verifying performance**
- **Tracking performance**

3 WELL-DEFINED

- **Defining a standard process**
- **Perform the defined process**
- **Coordinate practices**

4 QUANTITATIVELY CONTROLLED

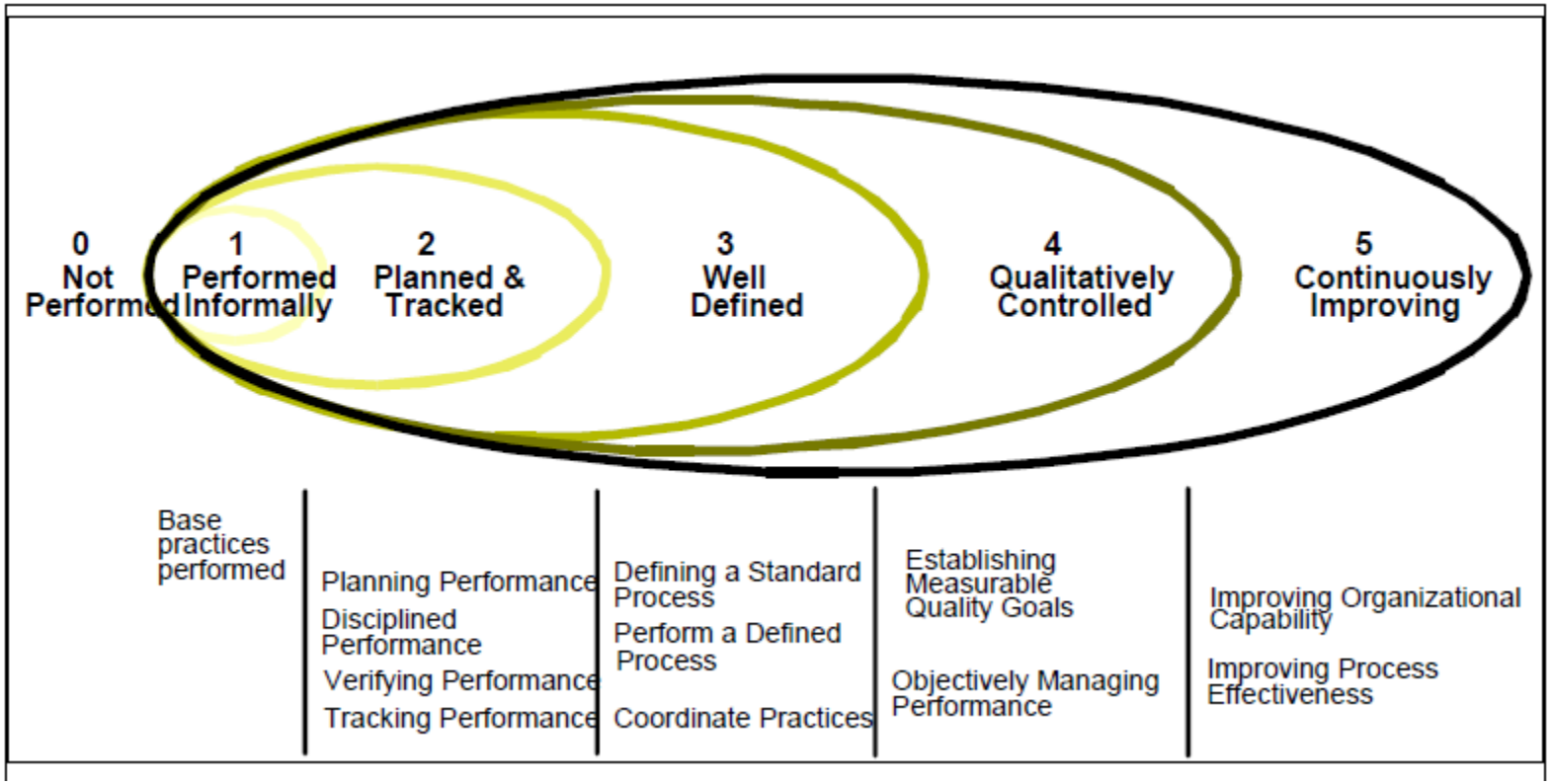
- **Establishing measurable quality goals**
- **Objectively managing performance**

5 CONTINUOUSLY IMPROVING

- **Improving organizational capability**
- **Improving process effectiveness**

Capability Levels

- Capability levels represent the maturity of orgs



Generic Practices

- Organized by capability level
  - I.e., add additional generic practices at each level
- Each level decomposed into set of common features
- Each set of common features consists of set of generic practices

Capability Level 1

- Common Feature 1.1 - Base Practices Are Performed
- GP 1.1.1 - Perform the Process

Capability Level 2

- Common Features:
  - Common Feature 2.1 - Planning Performance
  - Common Feature 2.2 - Disciplined Performance
  - Common Feature 2.3 - Verifying Performance
  - Common Feature 2.4 - Tracking Performance

Capability Level 2 Common Feature 1

- Focuses on aspects of planning to perform the Process Area and its associated Base Practices
- Generic Practices:
  - GP 2.1.1 - Allocate Resources
  - GP 2.1.2 - Assign Responsibilities
  - GP 2.1.3 - Document the Process
  - GP 2.1.4 - Provide Tools
  - GP 2.1.5 - Ensure Training
  - GP 2.1.6 - Plan the Process

Capability Level 3

- Common Feature 3.1 - Defining a Standard Process
  - GP 3.1.1 - Standardize the Process
  - GP 3.1.2 - Tailor the Standard Process
- Common Feature 3.2 - Perform the Defined Process
  - GP 3.2.1 - Use a Well-Defined Process
  - GP 3.2.2 - Perform Defect Reviews
  - GP 3.2.3 Use Well-Defined Data
- Common Feature 3.3 - Coordinate Practices
  - GP 3.3.1 - Perform Intra-Group Coordination
  - GP 3.3.2 - Perform Inter-Group Coordination
  - GP 3.3.3 Perform External Coordination

Capability Level 4

- Common Feature 4.1 - Establishing Measurable Quality Goals
  - GP 4.1.1 - Establish Quality Goals
- Common Feature 4.2 - Objectively Managing Performance
  - GP 4.2.1 - Determine Process Capability
  - GP 4.2.2 - Use Process Capability

Capability Level 5

- Common Feature 5.1 - Improving Organizational Capability
  - GP 5.1.1 - Establish Process Effectiveness Goals
  - GP 5.1.2 - Continuously Improve the Standard Process
- Common Feature 5.2 - Improving Process Effectiveness
  - GP 5.2.1 - Perform Causal Analysis
  - GP 5.2.2 - Eliminate Defect Causes
  - GP 5.2.3 - Continuously Improve the Defined Process

Limits of Capability Maturity Models

- Doesn't consider security policy (e.g., no MAC)
- Measures existence of process, but not quality
  - Existence is (roughly) objective – measurement is possible
  - Quality is subjective – no standard for comparison
- Doesn't guarantee good results
  - Does not measure effectiveness of processes
  - Can do everything but achieve nothing
- Non-uniformity of appraisals
- Misunderstanding of model and its use
  - Doesn't replace testing/evaluation
- Doesn't consider subversion

Build Security In Maturity Model (BSIMM)

- On-going project (http://www.bsimm.com/)
- Collected practices observed at real-world places ("67 leading software security initiatives" at leading companies)
- Compare target against comparison set
  - "Here's what everybody else is doing"

Adobe
Aetna
Bank of America
Box
Capital One
Citi
Comerica Bank
EMC
Epsilon
F-Secure
Fannie Mae
Fidelity
Goldman Sachs
HSBC
Intel
Intuit
JPMorgan Chase & Co.
Lender Processing Services Inc.
Marks and Spencer
Mashery
McAfee
McKesson
Microsoft
NetSuite
Neustar
Nokia
Nokia Siemens Networks
PayPal
Pearson Learning Technologies
QUALCOMM
Rackspace
Salesforce
Sallie Mae
SAP
Sony Mobile
Standard Life
SWIFT
Symantec
Telecom Italia
Thomson Reuters
TomTom
T. Rowe Price
Vanguard
Visa
VMware
Wells Fargo
Zynga

BSIMM Domains

- *Governance*: organization, management, and measurement of a software security initiative
- *Intelligence*: Collection of “corporate security knowledge”
- *SDL Touchpoints*: analysis and assurance of software development artifacts and processes
- *Deployment*: software configuration, maintenance, and other environment issues that have direct impact on software security

BSIMM Practices

- 112 activities organized into 12 practices in 4 domains
  - Activities organized into increasing level of

Governance	Intelligence	SDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

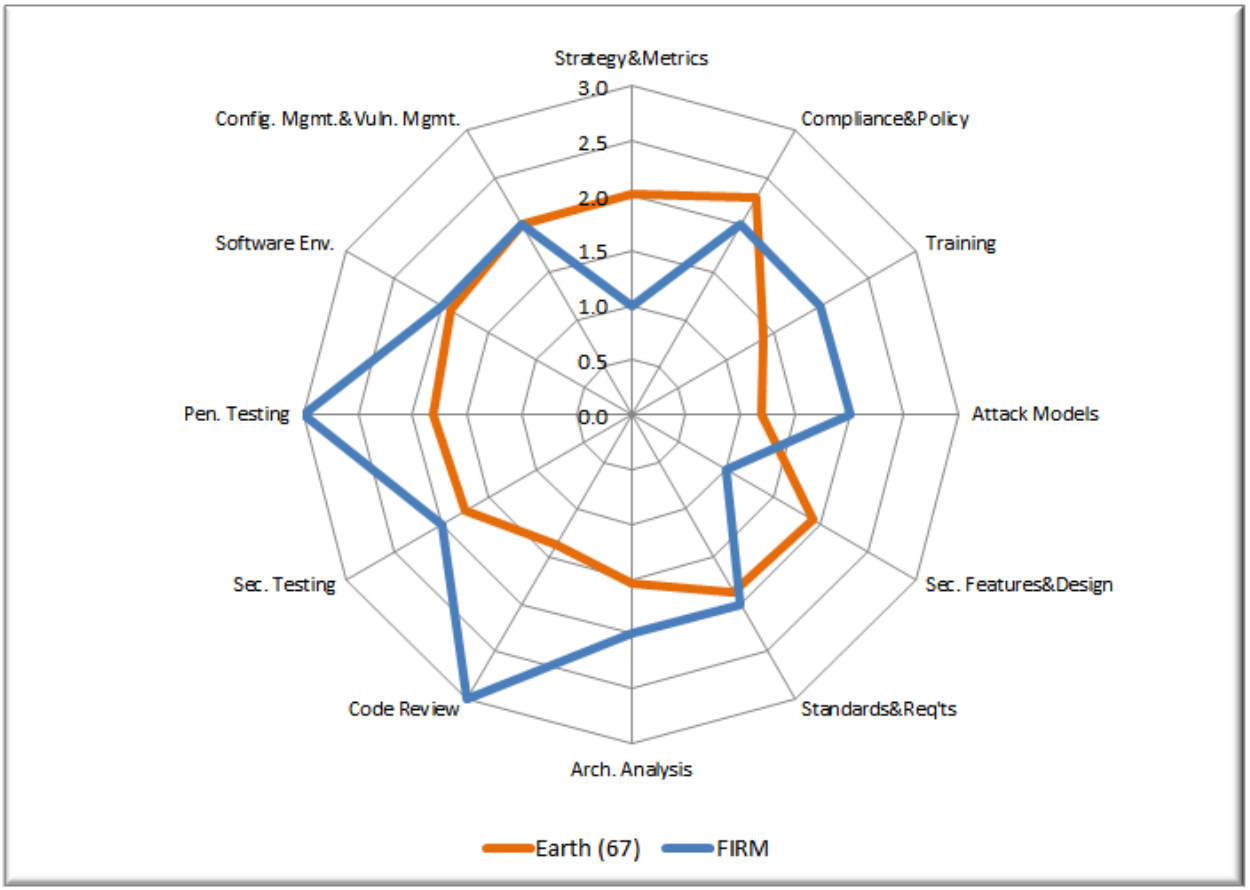
Example Practice: Architecture Analysis

Capturing software architecture diagrams, applying lists of risks and threats, adopting a process for review, building an assessment and remediation plan.			
Activity ID	Objective	Activity	Level
AA1.1	get started with AA	perform security feature review	1
AA1.2	demonstrate value of AA with real data	perform design review for high-risk applications	
AA1.3	build internal capability on security architecture	have SSG lead review efforts	
AA1.4	have a lightweight approach to risk classification and prioritization	use a risk questionnaire to rank applications	
AA2.1	model objects	define and use AA process	2
AA2.2	promote a common language for describing architecture	standardize architectural descriptions (including data flow)	
AA2.3	build capability organization-wide	make SSG available as AA resource or mentor	
AA3.1	build capabilities organization-wide	have software architects lead review efforts	3
AA3.2	build proactive security architecture	drive analysis results into standard architectural patterns	

Example Activity: AA1.1

- **Perform security feature review**
  - Identify security features in an application (authentication, access control, use of cryptography, etc.)
  - Look for problems that would cause these features to fail at their purpose or otherwise prove insufficient
- Example: “a system that was subject to escalation of privilege attacks because of broken access control would both be identified in this kind of review.”
- Where is the policy mentioned?

BSIMM Activity Coverage Graph



Source: http://www.bsimm.com/community/

Limits of BSIMM

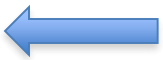
- Divorced from consideration of security policy
- Ad hoc. “Common practices” are not necessarily complete or even good
- Notes existence of process, but not quality
- No coding standards, except for reusing “mature” and “secure-by-design” frameworks
- Doesn’t consider subversion
  - Version management?

Microsoft Security Development Lifecycle (SDL)

- Mandated for use at Microsoft since 2004
- 7 phases:
  - Training
  - Requirements
  - Design
  - Implementation
  - Verification
  - Release
  - Response

Training Phase

- For all developers
- Major topics:
  - Common excuses for not fixing bugs
  - Secure design - identify and avoid issues that can lead to compromise (does this imply there is a policy?)
  - Threat modeling
  - Secure coding
  - Security testing
  - Privacy best practices



At last, a policy?

Requirements Phase

- Establish security and privacy requirements
  - Based on what criteria?
- Create quality gates/bug bars
  - E.g., require fixing of all “critical” vulnerabilities before release
- Perform security and privacy risk assessments
  - Determine need for threat modeling and security design reviews of components
  - Based on costs and regulatory requirements

Design Phase

- Establish design requirements
  - Validate all design specifications against a functional specification
  - Presumably, security functions identified in previous phase
- Perform attack surface analysis and reduction
  - Disable or restrict access to services
  - Reduce privilege
  - Layered defenses
- Use threat modeling
  - Microsoft STRIDE approach
  - We'll look at that in a future lecture

Implementation Phase

- Use approved tools
  - Approved compilers and linkers (and associated options and warnings)
- Deprecate unsafe functions and APIs
  - Ban unsafe functions
    - e.g., *gets* does not check bounds; use *fgets* instead
  - Use newer header files, compilers, or code scanning tools
- Perform static analysis
  - Automated tool scans code for common flaws without executing the code
  - We'll look at this in a future lecture

Verification Phase

- Perform Dynamic Analysis
  - Run-time verification of software functionality
  - Checks for memory corruption, user privilege issues, etc.
  - We'll discuss this topic in a future lecture
- Perform Fuzz Testing
  - Try to induce failure through malformed or random input data
- Conduct attack surface review
  - As in design phase, but on system basis
  - Tool takes snapshot of Windows system state before and after installation of product (http://www.microsoft.com/en-us/download/details.aspx?id=24487)

Attack Surface

- *Attack surface*: Exposed parts of systems that may have exploitable vulnerabilities, e.g.,
  - Open ports on outward facing web and other servers
  - Code that processes incoming data
  - Employees who can be "socially-engineered"
- Reference monitor (RM) abstraction helps
  - TCB shrinks attack surface
  - But still must consider threats to RM
- May have low-assurance components, not RM
  - E.g., your typical corporate IT network
  - Can still try to show non-bypassable, tamper-proof, and make assurance arguments
  - Security mechanisms probably have vulnerabilities!

Release Phase

- Create an Incident Response Plan
  - Includes emergency contacts and maintenance plan
  - For incidents with both internally developed and licensed software
- Conduct final security review
  - Examine artifacts of security activities (e.g., threat models) against quality gates/bug bars
- Certify release and archive
  - Attest that all security and privacy requirements met
  - Archive all specs, source, binaries, SDL artifacts, etc.

Limits of Microsoft SDL

- Still little or no connection to policy
  - Policy is implicit, perhaps
  - How do you know you have identified all (well, most) threats and requirements?
- Does it work? (http://www.gfi.com/blog/report-most-vulnerable-operating-systems-and-applications-in-2013/)
  - Application in 2013 with the most critical flaws discovered: Microsoft Internet Explorer
  - OS in 2013 with the most critical flaws discovered: Microsoft Windows Server 2008
    - Next 5: Windows 7, Vista, XP, Windows Server 2003, Windows 8

Why Is Microsoft Software Still so Vulnerable?

- Different targets
  - TCSEC: TCB that enforces a policy
  - MSDL: Any software; no policy
- Different requirements
  - TCSEC: Only goal is enforcing security policy
  - MSDL: “Security and privacy” requirements possibly secondary to other requirements
    - E.g., legacy code components and support
- Different testing
  - TCSEC: Think like an attacker – directed and prioritized
  - MSDL: Passive search for flaws – random, flat

Some Other Models

- SAFECode "Fundamental Practices for Secure Software Development"
  - http://www.safecode.org/publication/SAFECode_Dev_Practices0211.pdf
  - Aimed at reducing software weaknesses
- OWASP Software Assurance Maturity Model (OpenSAMM)
  https://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model
  - Many similarities to BSIMM – e.g., 4 domains, 12 practices
  - But more prescriptive focus
- Dept. of Homeland Security "Build Security In" initiative https://buildsecurityin.us-cert.gov/

NIST 800-171

- Some assessment methodologies are checklists of best practices that an organization should follow.
  - NIST SP 800-171: Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations
    - Lists 110 requirements in 14 areas (sometimes called families)

Access Control	Media Protection
Awareness & Training	Personnel Security
Audit & Accountability	Physical Protection
Configuration Management	Risk Assessment
Identification & Authentication	Security Assessment
Incident Response	System & Com Protection
Maintenance	System & Info Integrity

  - What is being assessed – An organization, end user, their systems (as deployed).

Least Privilege

Limit information system access to the types of transactions and functions that authorized users are permitted to execute.

SECURITY REQUIREMENTS	NIST SP 800-53 *Relevant Security Controls*		ISO/IEC 27001 *Relevant Security Controls*	
3.1 ACCESS CONTROL				
Basic Security Requirements				
3.1.1 Limit system access to authorized users, processes acting on behalf of authorized users, or devices (including other systems). 3.1.2 Limit system access to the types of transactions and functions that authorized users are permitted to execute.	AC-2	Account Management	A.9.2.1	User registration and de-registration
			A.9.2.2	User access provisioning
			A.9.2.3	Management of privileged access rights
			A.9.2.5	Review of user access rights
			A.9.2.6	Removal or adjustment of access rights
	AC-3	Access Enforcement	A.6.2.2	Teleworking
			A.9.1.2	Access to networks and network services
			A.9.4.1	Information access restriction
			A.9.4.4	Use of privileged utility programs
			A.9.4.5	Access control to program source code
			A.13.1.1	Network controls
			A.14.1.2	Securing application services on public networks
			A.14.1.3	Protecting application services transactions
			A.18.1.3	Protection of records
	AC-17	Remote Access	A.6.2.1	Mobile device policy
			A.6.2.2	Teleworking
			A.13.1.1	Network controls
			A.13.2.1	Information transfer policies and procedures
			A.14.1.2	Securing application services on public networks
Derived Security Requirements				

More on Authentication

Derived Security Requirements					
3.5.3 Use multifactor authentication for local and network access to privileged accounts and for network access to non-privileged accounts.	IA-2(1)	Identification and Authentication (Organizational Users) *Network Access to Privileged Accounts*	*No direct mapping.*		
	IA-2(2)	Identification and Authentication (Organizational Users) *Network Access to Non-Privileged Accounts*	*No direct mapping.*		
	IA-2(3)	Identification and Authentication (Organizational Users) *Local Access to Privileged Accounts*	*No direct mapping.*		
3.5.4 Employ replay-resistant authentication mechanisms for network access to privileged and non-privileged accounts.	IA-2(8)	Identification and Authentication (Organizational Users) *Network Access to Privileged Accounts-Replay Resistant*	*No direct mapping.*		
	IA-2(9)	Identification and Authentication (Organizational Users) *Network Access to Non-Privileged Accounts-Replay Resistant*	*No direct mapping.*		
3.5.5 Prevent reuse of identifiers for a defined period.	IA-4	Identifier Management	A.9.2.1	User regi de-regist	

SECURITY REQUIREMENTS	NIST SP 800-53 *Relevant Security Controls*		ISO/IEC 27001 *Relevant Security Controls*	
3.5.6 Disable identifiers after a defined period of inactivity.	IA-4	Identifier Management	A.9.2.1	User registration and de-registration
3.5.7 Enforce a minimum password complexity and change of characters when new passwords are created.	IA-5(1)	Authenticator Management *Password-Based Authentication*	*No direct mapping.*	
3.5.8 Prohibit password reuse for a specified number of generations.				
3.5.9 Allow temporary password use for system logons with an immediate change to a permanent password.				
3.5.10 Store and transmit only cryptographically-protected passwords.				
3.5.11 Obscure feedback of authentication information.	IA-6	Authenticator Feedback	A.9.4.2	Secure logon procedures

New Models for Assessment

- There are new assessment methodologies in the works that combine the checklist of controls (e.g. NIST 800-171) with maturity assessment approaches (e.g. CMMI Capability Maturity Model Integration).
  - US DoD Cybersecurity Maturity Model Certification (CMMC)
    - As envisioned levels correspond to the subset of controls implemented AND the highest level of maturity achived.
    - 1 is Basic hygiene, perhaps 17 controls, and ad hoc security processes
    - Level 2 adds controls and required security processed to be documented
    - Level 3 adds more controls and required processes guided by policy
    - Level 5 reqires continuous improvement of process and sharing of data.

Reading Next Lecture

The following readings are linked from the class web page: (ccss.usc.edu/523)

- Attack Trees
- *A Requires/Provides Model for Computer Attacks*
- *Uncover Security Design Flaws Using the STRIDE Approach*
- *Foundations of Attack–Defense Trees*
- *Threat Risk Analysis for Cloud Security based on Attack-Defense Trees*