ENPM 809W
Introduction to Secure Software Engineering

Gananand Kini

Lecture 1

Introduction to the course

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

Instructor

- **Mr. Gananand Kini**
- **Lead Cybersecurity Engineer at MITRE Corporation**
- **Email: gkini@umd.edu**

- **Please use ELM's email system for contacting me about the course.**

- **Please Note: The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author.**

All materials are licensed under a Creative Commons "Share Alike" license

- http://creativecommons.org/licenses/by-sa/3.0/

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

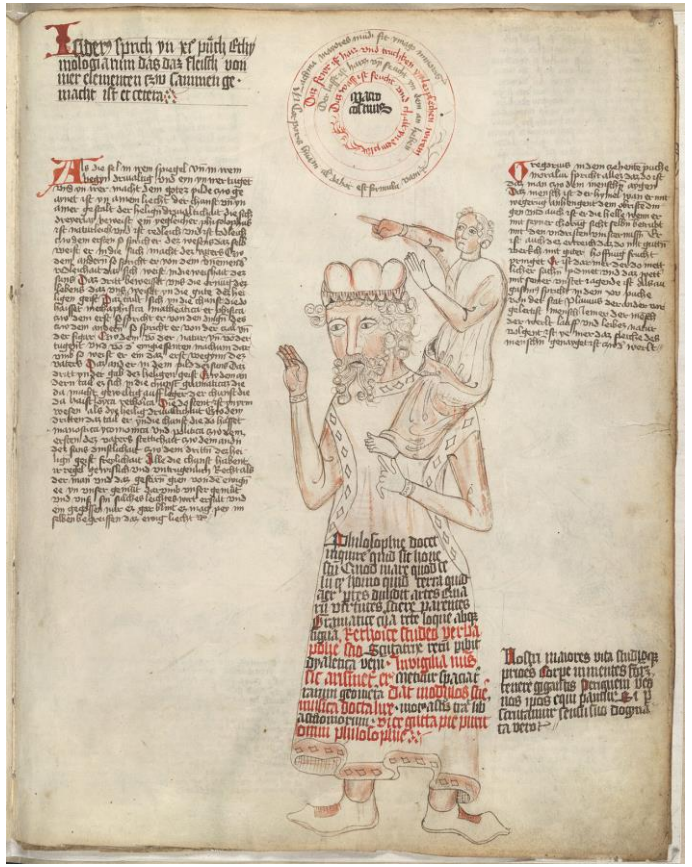
Attribution condition: You must indicate that this is derivative work 'Is derived from Introduction to Secure Software Engineering class by Gananand Kini et al.'

We stand on the shoulders of giants…

The materials in this course are derived from the courses as shown below. Thank you to all the authors below.

- **Title:** Secure Software Design and Programming: Class Materials
- **Author:** Prof. David A. Wheeler
- **Source:** https://dwheeler.com/secure-class/index.html
- **License:** CC BY-SA 3.0

- **Title:** Introduction to Secure Coding
- **Author:** Andrew Buttner and Larry Shields
- **Source:** http://OpenSecurityTraining.info/IntroSecureCoding.html
- **License:** CC BY-SA 3.0



By Unknown author - http://lccn.loc.gov/50041709, Public Domain, https://commons.wikimedia.org/w/index.php?curid=14901168

Outline

- Course syllabus
- Why is most software insecure?
- Why is it important to implement secure software?
- Security Development Lifecycle

Course Syllabus

Learning Objectives

- Define and apply the Secure Development Lifecycle (SDLC).
- Understand how to define, communicate and mitigate software security weaknesses as part of the SDLC.
- Apply penetration testing techniques to test software security.
- Apply security analysis techniques and tools to assess software security.
- Be able to explain weaknesses in software systems.
- Apply defenses to remediate software exploits and weaknesses.

What this course is NOT about…

- Antivirus technology
- Hacking and advanced exploitation
- Software Design and Architecture (may highlight relevant points but not the main objective)

Students are expected to …

- Refer to the syllabus.
- Perform assigned reading.
- Complete and submit homework according to schedule.
- Perform in-class laboratories and write a lab report to be turned in the following class.
- Attend all lectures, ask plenty of questions.
- Complete homework, quizzes, exam and a project.
- Provide proper citation and references in all their work.
- Discuss in ELMs discussion groups with any questions about class material.

Grading Criteria

▪ Lab Reports	20%	A	90 - 100
▪ Homework	15%	B	78 - 89
▪ Quizzes	15%	C	60 - 77
▪ Final Exam	25%	D	50 - 59
▪ Project	25%	F	0 - 49

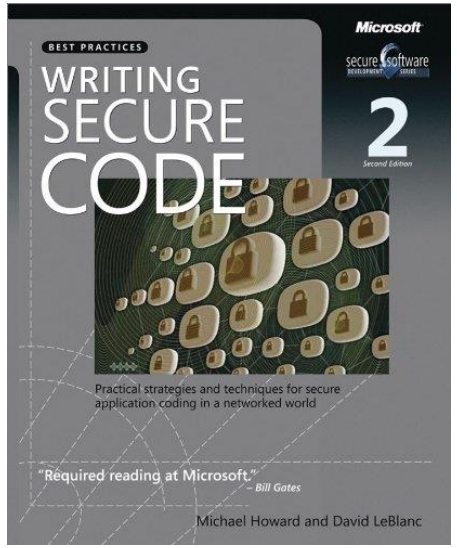
- Late submissions will be penalized at 10% per day (ending midnight EST).
- Absolute Grading – your performance will be reflected by the amount of work YOU put into the course.
- No extra credit will be awarded in this course.

From this course, you will gain…

- **The ability to audit a software system and find security weaknesses.**
- **Describe the weaknesses you find using MITRE's CWE™.**
- **Methodologies, techniques and tools used in a secure code review.**
- **Methodologies, techniques and tools used in finding security weaknesses.**
- **Techniques and tools to fix the found weaknesses.**
- **How to perform secure code reviews:**
  - For finding security weaknesses
  - For improving code quality
  - For improved communication about code content and securing software
  - For educating others

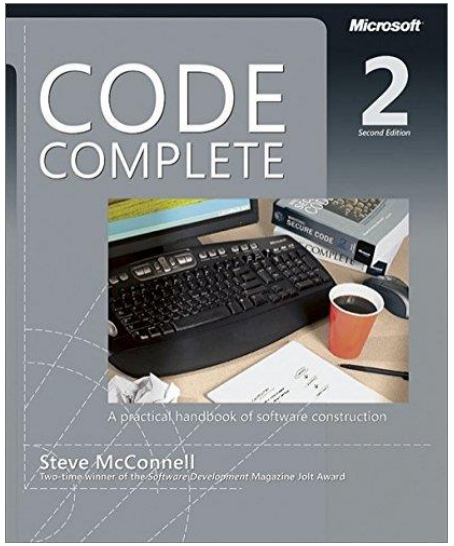
Relevant Books

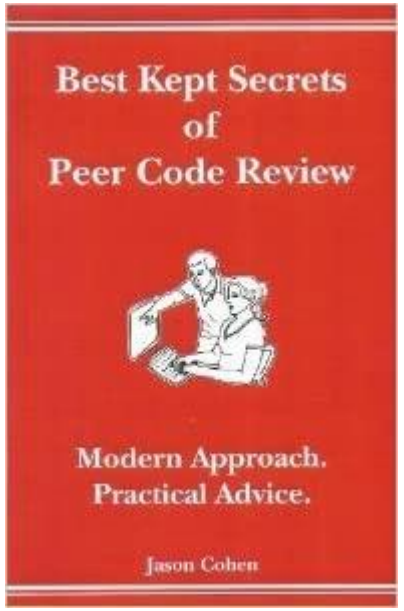
- These are for reference only and NOT REQUIRED for the course.
- Secure Programming HOWTO – Creating Secure Software by David A. Wheeler (available free at https://dwheeler.com/secure-programs/)
- Writing Secure Code (Developer Best Practices) 2nd Edition by David LeBlanc and Michael Howard



Relevant Books contd…

- These books don’t necessarily cover security and are focused more on bugs, however a lot of the practices are applicable.
- Code Complete (Developer Best Practices) 2nd Edition by Steve McConnell from Microsoft Press. ASIN: B00JDMPOSY
- Best Kept Secrets of Peer Code Review by Jason Cohen. ISBN-10: 1599160676, ISBN-13: 978-1599160672





Why this course…

- A lot of security education today is focused on attacking systems to show what is bad…
- Instead of focusing on what you can do to secure the software system with good practices.
- Software engineering classes focus primarily on design, software requirements and implementation with no time to dedicate to security.
- There are a few examples of this changing currently in academic curriculums.

Course Organization

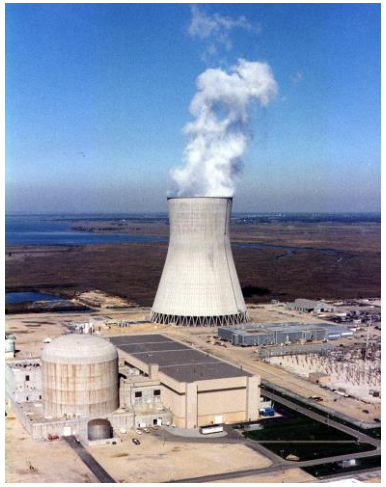
- **This course goes into:**
  - Tenets of Software System Design
  - Security Principles and the Secure Software Development Cycle
- **After these, the weeks are divided into attack and defense of different parts of an application:**
  - Input Security
  - Cryptography
  - Authentication & Authorization
  - Session Management
  - Error Handling & Logging & Debug Code
  - Performing Secure Code Reviews

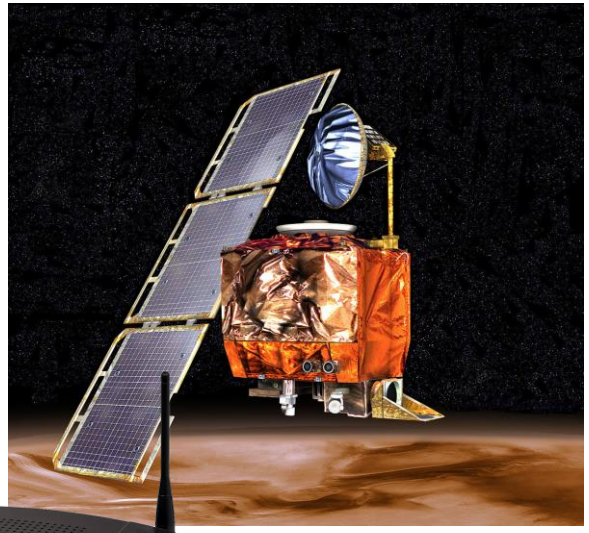
Course Details & Important Dates

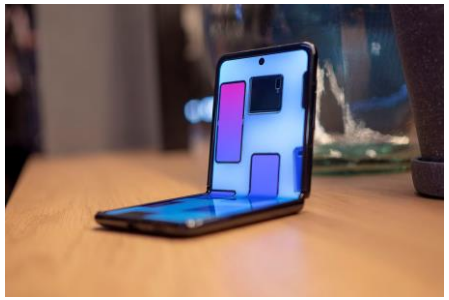
- Class meets Tuesdays 4 – 6:40 pm
- First Class: August 31st, 2021
- Project Phase 1 Due: September 14th, 2021
- Project Phase 2 Due: November 9th, 2021
- Project Phase 3 Due: November 30th, 2021
- Project Presentations: Either starting November 30th, 2021 or December 7th, 2021 (depending on number of students)
- Final Exam: December 21st, 2021 or 12/21/21

Why is most software insecure?

Which of these run software?



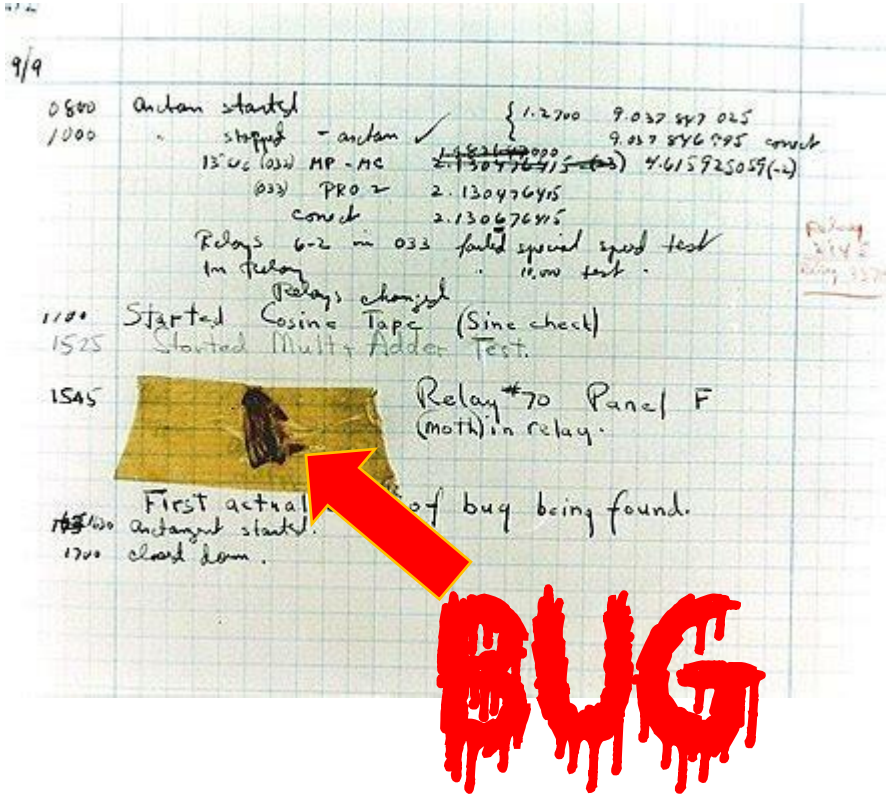




Clearly…

- **Software plays a major role.**
- **A lot of resources are focused on making the software work.**
- **But not enough attention is paid to:**
  - What could make the software NOT be safe …
  - What could make the software NOT be secure …
  - What could make the software NOT behave correctly…
  - What could make the software NOT have effects on other parts of the system or systems …
  - … and still sometimes the software does NOT work. Why?

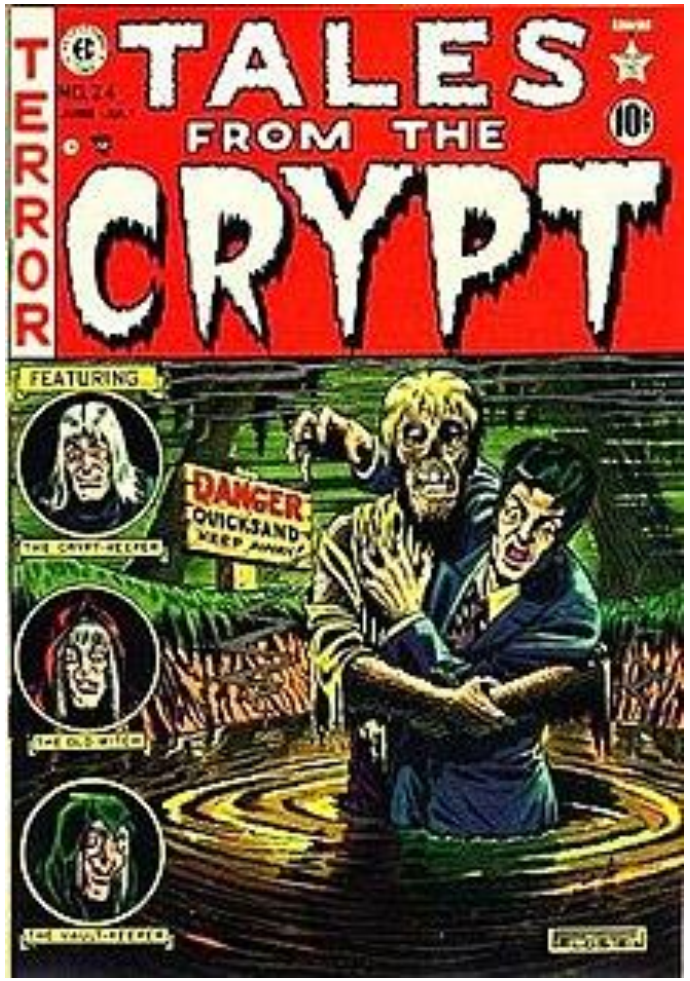
Rear Admiral Grace M. Hopper coined the term …



Source: *SI Neg. 83-14878. Date: na.*..Grace Murray Hopper at the UNIVAC keyboard, c. 1960. Grace Brewster Murray: American mathematician and rear admiral in the U.S. Navy who was a pioneer in developing computer technology, helping to devise UNIVAC I. the first commercial electronic computer, and naval applications for COBOL (common-business-oriented language). ..Credit: Unknown (Smithsonian Institution)
And en.wikiquote.org

Some of the reasons for insecure software…

- Poor communications (between all parties involved)
- Commercial pressures
- Poor system design
- Poorly defined system requirements
- Inaccurate estimates of required resources
- Unmanaged Risks
- Sloppy Development Practices
- Use of immature technologies
- Stakeholder politics
- …



Sources:
1. Why Software Fails - IEEE Spectrum (2005)
2. https://en.wikipedia.org/wiki/File:Tales_from_the_Crypt_24.jpg

Why is it important to implement secure software systems?

Repercussions of Software Failures

- **Injury & Loss of Life [1]**
- **Loss of financial assets [2]**
- **Loss of resources: organizational assets [3], the organization itself [2].**
- **Loss of the software system: Denial of service**
- **Loss of valuable or sensitive information**
- **Loss of trust***

YEAR	COMPANY	OUTCOME (COSTS IN US $)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to $33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to $3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after $54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately $400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing $527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to $160 million loss.
2003–04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of $100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after $170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after $33.2 million† is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to $445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to $100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after $130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing $25 million.
1999	United Way	Administrative processing system canceled after $12 million is spent.
1999	State of Mississippi	Tax system canceled after $11.2 million is spent; state receives $185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to $151 million loss.
1998	Snap-on Inc.	Problems with order-entry system contribute to revenue loss of $50 million.
1997	U.S. Internal Revenue Service	Tax modernization effort canceled after $4 billion is spent.
1997	State of Washington	Department of Motor Vehicle (DMV) system canceled after $40 million is spent.
1997	Oxford Health Plans Inc.	Billing and claims system problems contribute to quarterly loss; stock plummets, leading to $3.4 billion loss in corporate value.
1996	Arianespace [France]	Software specification and design errors cause $350 million Ariane 5 rocket to explode.
1996	FoxMeyer Drug Co.	$40 million ERP system abandoned after deployment, forcing company into bankruptcy.
1995	Toronto Stock Exchange [Canada]	Electronic trading system canceled after $25.5 million** is spent.
1994	U.S. Federal Aviation Administration	Advanced Automation System canceled after $2.6 billion is spent.
1994	State of California	DMV system canceled after $44 million is spent.
1994	Chemical Bank	Software error causes a total of $15 million to be deducted from 100 000 customer accounts.
1993	London Stock Exchange [UK]	Taurus stock settlement system canceled after $600 million** is spent.
1993	Allstate Insurance Co.	Office automation system abandoned after deployment, costing $130 million.
1993	London Ambulance Service [UK]	Dispatch system canceled in 1990 at $11.25 million**; second attempt abandoned after deployment, costing $15 million.**
1993	Greyhound Lines Inc.	Bus reservation system crashes repeatedly upon introduction, contributing to revenue loss of $61 million.
1992	Budget Rent-A-Car, Hilton Hotels, Marriott International, and AMR [American Airlines]	Travel reservation system canceled after $165 million is spent.

Sources:

1. Medical Devices: The Therac-25 (1995) by Nancy Leveson
2. Why Software Fails - IEEE Spectrum (2005)
3. Analyzing software failure on the NASA Mars Climate Orbiter (2017) by Christopher Demicoli

And if that wasn't bad enough … Adversaries

- **Dedicated attackers looking to undermine such software systems by exploiting vulnerabilities in those systems who are :**
  - Well organized – almost like a small business
  - Well resourced – can range from organized crime to nation states
  - Sophisticated – technically and in means of attack

Script Kiddie
- Leverages tools and exploits created by others
- Hacking by pushing buttons

Hacktivist
- Hacker with a cause
- Denial of service, site defacement

Hacker
- Malicious and non-malicious
- Motivated by ability to hack

Cyber Criminal
- Different levels of sophistication
- Organized
- Scams, information theft, fraud

Advanced Persistent Threat
- Extremely sophisticated
- Try to stay hidden
- Slow and methodical
- Information Theft & Espionage

Identifying risks means looking at Attacks

- **Attacks have been changing …**
  - Started with attacking operating systems and networks
  - Changed to web applications, browsers and mobile applications
  - Now includes embedded platform software including Internet-of-Things (IoT), medical devices etc.
- **This means your software will be attacked too… Guaranteed!**
  - How many times have you updated Windows? Android? iOS?
  - Now how about your washer/dryer?
  - How about your electric meter?

Sources:
1. https://www.comics.org/issue/13047/

What can we do about insecure software systems?

We can address some of these reasons …

- **Poor System Design**
  - Think security in during system design
- **Unmanaged Risks**
  - Identify security risks beforehand
- **Sloppy Development Practices**
  - Put measures in to prevent such practices including but not limited to:
    - Software Development guidelines
    - Software Lifecycle management
    - Code Review Processes

“Security Features” are NOT the answer!

- Just add crypto …
- Just install security software …
- Just implement your own cryptographic algorithm …
- Just install anti-virus software …
- Just use open source software …

Security through obscurity is also NOT the answer…

- **Adding barriers can make users:**
  - Find ways around … making it even more insecure
  - Lose trust*
- **Sometimes feels like a tradeoff between security and usability or convenience**
- **However, progress is being made in usable security…**

Sources:
1. Image of snow tracks around gate. https://www.youtube.com/watch?v=pqyFG-G4hF8. Retrieved July 10, 2021.

Audience Poll

Are you familiar with any programming languages?

If so, Which of these languages (in no particular order) are you:

1. Most proficient?
2. Least proficient?
3. Can get along using?

.NET (C#, VB.NET etc.)
C/C++
Java/Scala etc.
Ruby
Swift
JavaScript / ECMAScript
Python
Go
PHP
Rust
Shell Scripting

C# .NET will be used in this class…

- If you want to know more, please see https://dotnet.microsoft.com/learn.
- Chosen because it has a lots of nice tools and IDEs available for free development and helps the learning experience.
- A lot of the security principles can be easily demonstrated in this language and represents a real life software system in use by a lot of organizations today.
- Similar to Java if you have used Java before and hopefully easier to learn.
- This is a class on software engineering and you are expected to learn and come up to speed with C# .NET and ASP .NET on your own!

C# .NET and skills applicability

- **Despite picking C# .NET, a virtual machine runtime based language, the skills you will gain in this class apply to a wide variety of languages, frameworks and software systems.**

- **Prof. David A. Wheeler at GMU even had one of his students get a pay raise for implementing some of these techniques!**

- **These skills are also very applicable inside and outside the classroom.**

How to measure security?

- **Lots of data showing defect rates (bugs) versus code complexity or size.**

- **You might think … No real datasets that show security defects in code.**

- **Lots of places to get vulnerability information from:**
  - https://us-cert.cisa.gov/ncas/alerts (US CERT)
  - https://securitytracker.com/topics/topics.html (Security Tracker)
  - https://nvd.nist.gov/vuln/full-listing (National Vulnerabilities Database/CVE)
  - https://www.securityfocus.com/vulnerabilities (Security Focus)

CWE™ – Common Weakness Enumeration by MITRE

- **The 2021 Top 25 Most Dangerous Software Weaknesses**
- **https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html**

- **The common weaknesses enumeration defines common weaknesses in software that have security implications.**

- **Different from vulnerabilities**
  - Weaknesses can be thought of as mistakes or flaws (intentional or unintentional) in the software that can have serious system level impacts.
  - Vulnerabilities allow adversaries to take advantage of ***accessible*** weaknesses that are ***exploitable*** in order to achieve their attack goals.
- **Some Common Vulnerabilities and Exposures (CVE™) entries, also by MITRE, reference CWEs.**

Weak spots in software

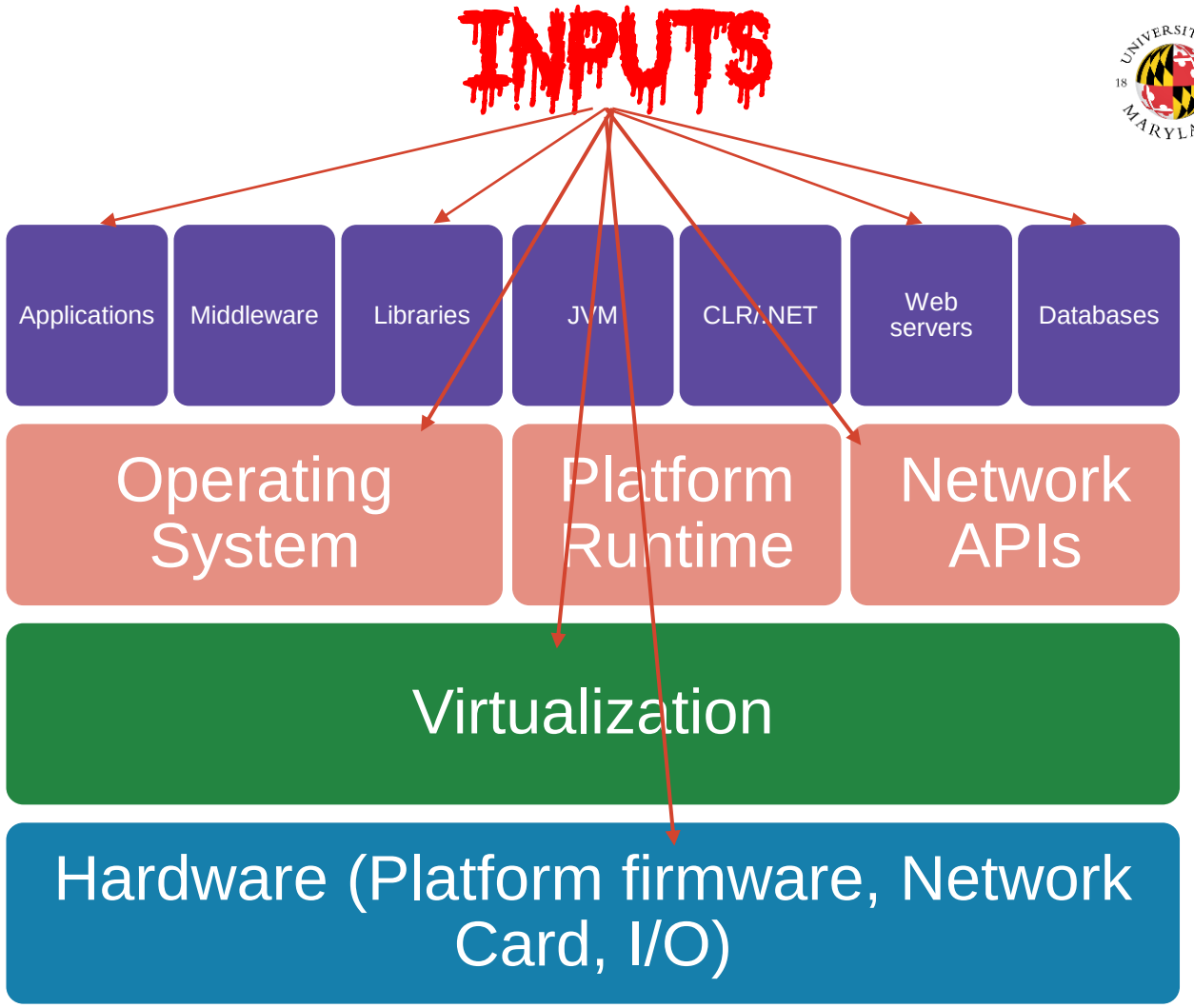
- As you can see from the CWE Top 25 Most Dangerous Weaknesses List (2021), input validation category of weaknesses take up the Top 6!
- Attacks typically take advantage of INPUTS to the system.
- Findings from Langsec (http://langsec.org/), security conference focused on Language-theoretic security

From: Occupy Babel

Sources:
1. http://langsec.org/occupy/

Software has layers…

- **Weaknesses can exist anywhere along these layers.**
- **As you go top down on this stack, security weaknesses become more critical in their impacts!**
- **As you go across, the software tends to present a more diverse surface at the top as compared to the bottom. E.g. all BIOS chipset firmware from a vendor as compared to a single desktop applications.**



Weaknesses in software can be introduced ….

- As you just saw, security weaknesses can be introduced at multiple levels.
- Can also be introduced along any of the phases in the system development lifecycle.
- CWEs similarly also exist for such weaknesses that can be introduced along different phases. E.g. CWE-1110: Incomplete Design Documentation.

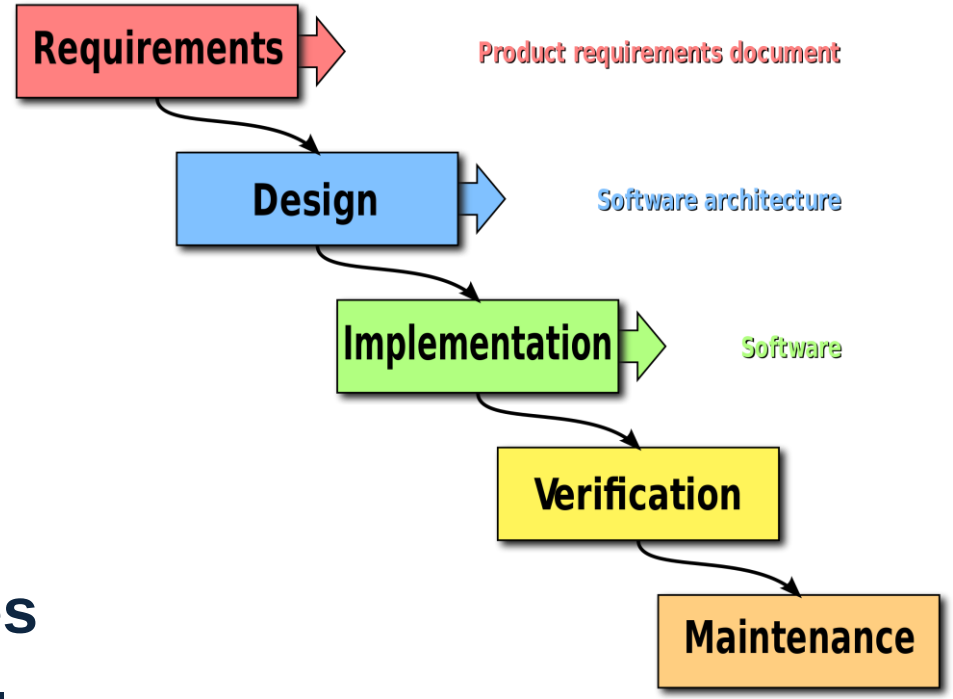


Figure 1. Typical Waterfall Model of System Development.

Sources:

1. https://upload.wikimedia.org/wikipedia/commons/thumb/e/e2/Waterfall_model.svg/1600px-Waterfall_model.svg.png Wikipedia by Peter Kemp, Paul Smith

That's great! Is it possible to write secure code at all?

- **Yes.**

- **Developers don't share how to write secure code… Just like…**
- "When we got started no one was talking about code review in the press, so we didn't think many people were doing it. But our experience has made it clear that peer code review is widespread at companies who are serious about code quality. But the techniques are still a secret! …" **- Jason Cohen from *Best Kept Secrets of Peer Code Review* (2006).**

- **CWE™ can help communicate such weaknesses and dispel such secrecy.**

- **This class is designed to also help.**

Security Development Lifecycle

Models for secure development

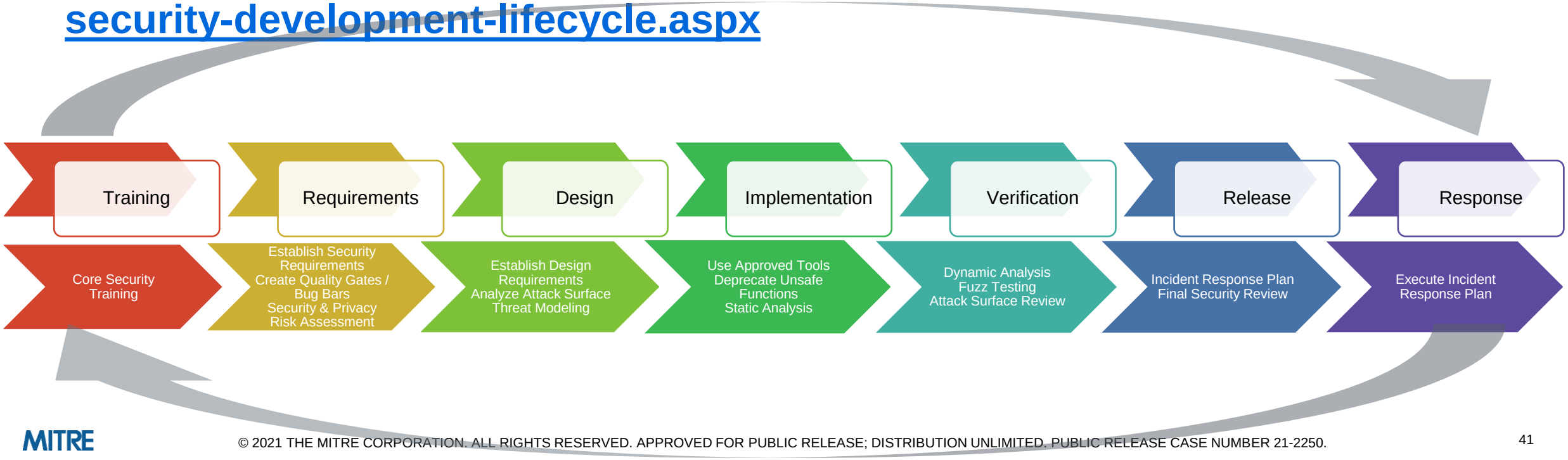
- **BSIMM – Building Security In – Maturity Model (https://www.bsimm.com/)**

- **OWASP SAMM – Formerly OpenSAMM or Open Software Assurance Maturity Model (https://owaspsamm.org/) is now part of OWASP. You can download the pdf.**

- **Microsoft's Secure Development Lifecycle (SDLC)**

Security Development Lifecycle

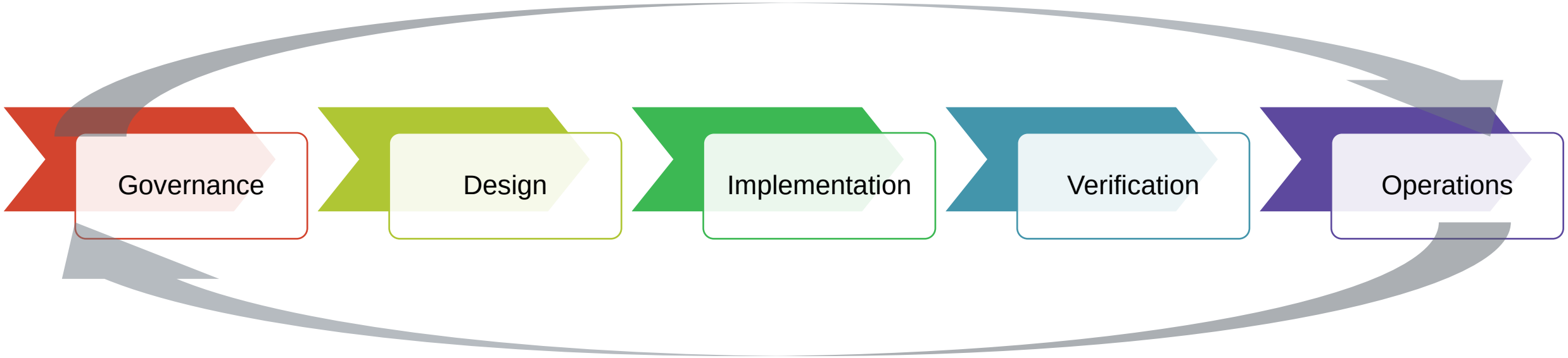
- **Luigi Bruno and Richard Mueller, Microsoft, Jan 2012**
- **Several phases and techniques to design, implement and maintain a software system.**
- **Found here: https://social.technet.microsoft.com/wiki/contents/articles/7100.the-security-development-lifecycle.aspx**



OWASP SAMM

- **Started as the Open SAMM project, but transitioned to OWASP to maintain.**

- **A lot of the techniques similar to the Microsoft SDLC.**



Governance and Education Phase – Security Training

- **Developer training is key, especially in secure code practices.**

- **Training that focuses on:**
  - Secure Design
  - Threat Modeling
  - Secure Coding
  - Security Testing
  - Privacy analysis
  - Policy and Governance guidelines

Requirements Phase

- This phase is used to establish the security requirements of the entire software system. A security risk assessment and a privacy risk assessment can help to identify which functions of the software need to be closely looked at.
- For example, if the software stores personal information, the functionality of the software that governs stores, transmission and uses should be examined closely and requirements such as HIPAA rules should be applied to determine what security requirements should be put in place.
- What is the minimum criteria that must be met in order to pass the quality or security of the software system for a release?
- For example, no vulnerabilities marked critical should be present in the release.
- Once set, should not be relaxed or compromised. Getting this agreement ahead of time from stakeholders can help establish the criteria.

Design Phase

- With the security requirements in hand, the design phase can proceed accounting for those additional security requirements. The design requirements outline the plan for the project to follow through the rest of the phases.

- The security requirements should not be thought of as an add-on. The security requirements should be “baked-in” to the design of the software system.

- Especially with cryptographic functionality, the design requirements should account for what, how and why the minimum cryptographic requirements should be used.

- Attack Surface analysis/reduction and Threat modeling is generally performed during this phase.

Implementation Phase

- The implementation phase of the security development lifecycle is focused on how the software system is planned to be developed, deployed and used.
- This includes all the documentation and tools geared towards the goal above.
- This phase generally involves identifying and using only an approved list of tools, security checks etc.
- It also involves deprecating unsafe functions of the software system and performing static code analysis.

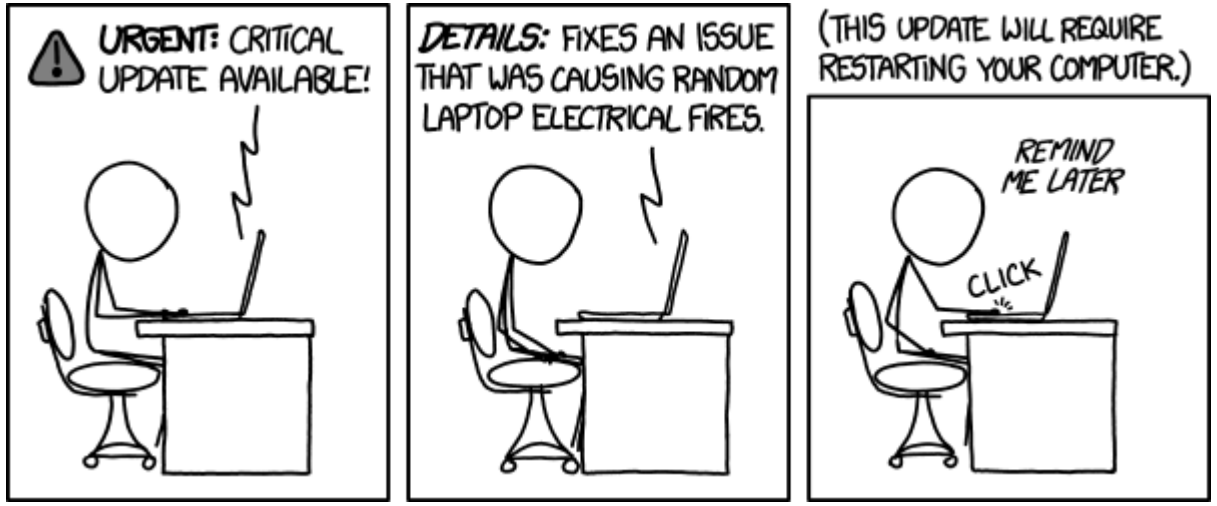
Verification Phase

- The verification phase acts as the phase where you are able to check and verify whether the security requirements were really met in the implementation.
- A public release security and privacy review can be performed during this phase. This may also include any consultations with institutional review boards.
- Dynamic Analysis techniques are used to test the software system as it runs.
- Fuzz Testing techniques are also used to ensure no failures due to random input result.
- An attack surface review is performed to ensure any and all attack vectors have been addressed as a result of any changes in the previous phases.

Operations Phase – Release and Incident Response

- **I am just a coder… Who cares after the code is released? #amiright**

- **This is a critical part of the lifecycle!**



- **The Release phase:**
  - Is used to ready the software system for consumption by customers.
  - A final security and privacy review is performed.
  - Licensing of the software and its terms and any servicing terms for third-party software are put in place.
  - Release archiving is performed storing all source code, documentation, specifications, design (including security) documents are archived.
- **An incident response plan is put in place including any Emergency response plans for the software system.**

Source:
1. XKCD: Update. xkcd: Update. Retrieved July 10, 2021.

Next time …

- **Software Engineering Design and Security Principles**