Software Programming Practices and Development

SlipperyRock University of Pennsylvania

Dr. Nitin Sukhija, Director C2AC and Associate Professor, Department of Computer Science

Secure Software Programming Practices and Development

Dr. Nitin Sukhija, Director C2AC and Associate Professor, Department of Computer Science

About me

Nitin Sukhija
Associate Professor and Director of Center for Cybersecurity and Advanced Computing (C2AC), LBNL affiliate, Campus Champion Leadership team, ACM SIGHPC education chapter
Department of Computer Science
Slippery Rock University of PA

High performance Data Analytics and Cyber Resilience
nitin.sukhija@sru.edu (or google)

Outline

- Introduction
- Challenges
- Software Development Life Cycle
- Secure Software Development Life Cycle
- Threat Modeling
- SAST and DAST tools
- Question and Answers

Why are we talking about Secure Software Development ?

Case 1: SolarWinds Attack 2020

- a supply chain attack leading to data breaches globally
- threat actors turned the Orion software into a weapon gaining access to several government systems and thousands of private systems around the world
- infected up to 18,000 customers globally, including major U.S government departments
- one of the most sophisticated cyberattacks in the software lifecycles ever deployed

Case 1: SolarWinds Attack 2020

- a supply chain attack leading to data breaches globally
- threat acto pon gaining d thous
- infecte uding major U.S governm
- one of the most so sticated cyberattacks in the software lifecycles ever deployed

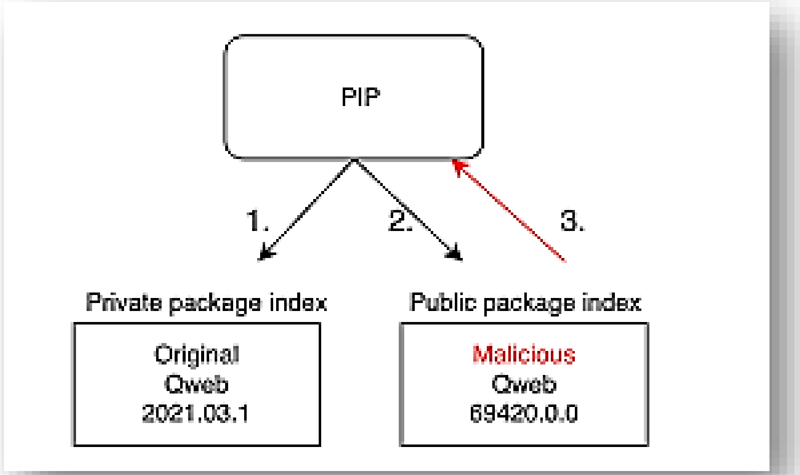
✓ not enough to build a firewall and hope it protects

✓ need to actively seek out vulnerabilities in systems software

solarwinds

Case 2: Dependency Confusion Attack 2021

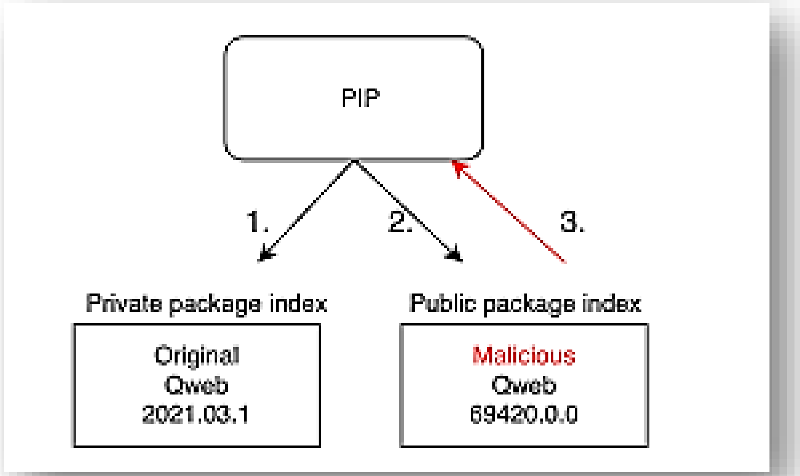
- unveiled by security researcher Alex Birsan (@alxbrsn) in 2021
- targets third-party dependencies.
- threat actors inject malicious code into the dependencies the application uses, allowing them to access the application and its data.
- breached the systems belonging to Microsoft, Apple, Uber, and Tesla.



Case 2: Dependency Confusion Attack 2021

- unveiled by security researcher Alex Birsan (@alxbrsn) in 2021
- targets
- threat dencies the a s the application
- breached the syst s belonging to Microsoft, Apple, Uber, and Tesla.

✓ developers must know about the risks of dependency confusion and the importance of secure package management



Case 3: NATO Data Breach 2022

- a data breach on the NATO's Communities of Interest (COI) Cooperation Portal

- software was injected with malware by SiegedSec (a cybercrime group with a history of politically-motived attacks)

- 3,000 stolen files total more than nine gigabytes of sensitive data intended for NATO countries and partners..

- was a response to NATO's human rights violations and because it was "fun to leak documents."

Case 3: NATO Data Breach 2022

- a data brea st (COI) Co
- soft cyb attac
- 3,000 of sensitive da d partners..
- was a response to O's human rights violations and because it was "fu o leak documents."

✓ Promote secure coding practices during the development process to minimize vulnerabilities that attackers could exploit to inject malware.
✓ Regularly conduct security code reviews and utilize automated security testing tools to identify potential weaknesses.

Case 4: MOVEit data breach Attack 2023

- a zero-day exploit of Progress Software's MOVEit Transfer enterprise file transfer tool
- used by a ransomware gang called Cl0p
- allowed the hackers to inject SQL commands and access the databases of MOVEit customers
- the biggest data theft of 2023
- over 2,500 organizations have reported being attacked, with data thefts affecting more than 62 million people as of Oct 2023

Case 4: MOVEit data breach Attack 2023

- a zero-day exploit of Progress Software's MOVEit Transfer enterprise file transfer tool

- used by a ransomware gan

- allowed the ha s the databas

- the biggest

- over 2,500 organizations ve being attacked, with data thefts affectin ore than 62 million people as of Oct 2023

✓ Check indicators of compromise (IoCs) that may suggest your software has been affected by the attack.

What’s really going on here ?

What’ here ?

Cost of Cyberattacks – Billions!!

Continued..

Challenges

Software Development Life Cycle

Secure Software Development Life Cycle

Threat Modeling

SAST and DAST tools

Question and Answers

Increasing Complexity and Threat Vectors

Number of unpredictable and highly dynamic factors

1)Heterogeneity of interconnected networks, applications, server, and virtualized infrastructure components

2) Proliferation in the hybrid system models and orchestration of many complex software services

3) Convergence of Big data, HPC and Cloud services

4) Reliance of more and more software applications on open-source packages and third-party dependencies

5) Dependence on digital data transfer becoming increasingly common for businesses of all sizes and domains.

Increasing Complexity and Threat Vectors

Number of unpredictable and highly dynamic factors

1)Heterogeneity of int ons, server, and virtualized infrastr

2) Proliferation ion of many complex softwa

3) Convergence of

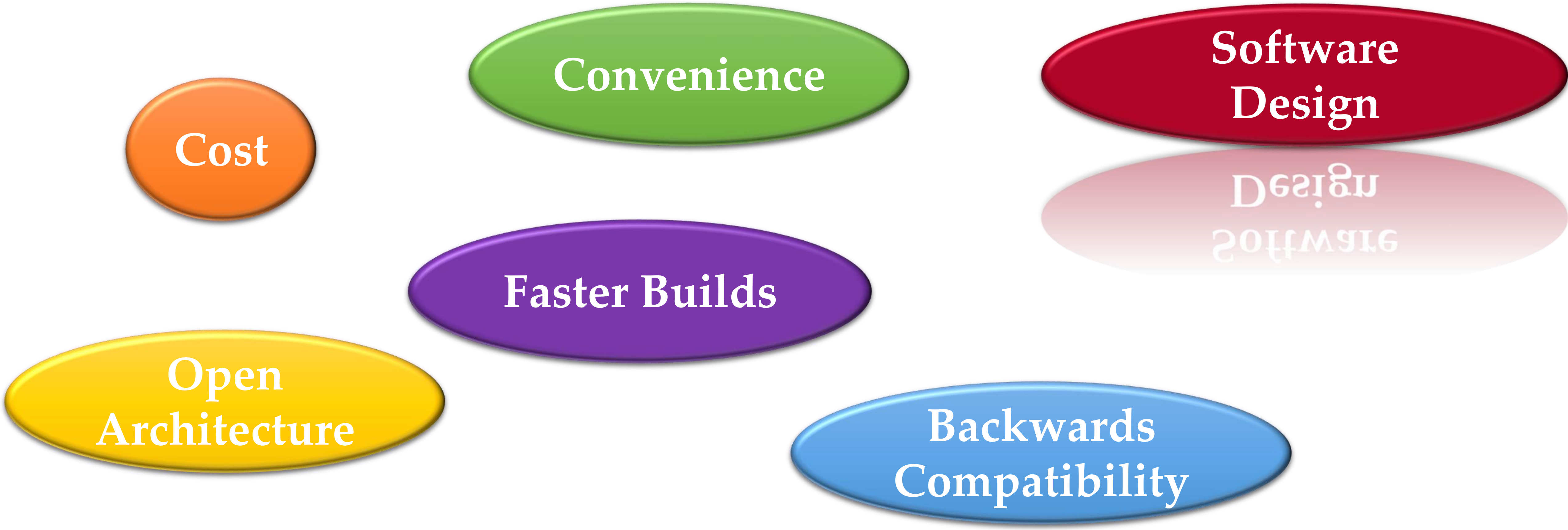
4) Reliance of more and more software applications on open-source packages and third-party dependencies

5) Dependence on digital data transfer becoming increasingly common for businesses of all sizes and domains.

- ✓ Risks continue to increase!
- ✓ Vendors are likely unaware that their software, apps or updates are infected with malicious code when released to the public

Security Not a Priority

- During Software Development other design priorities often trump security

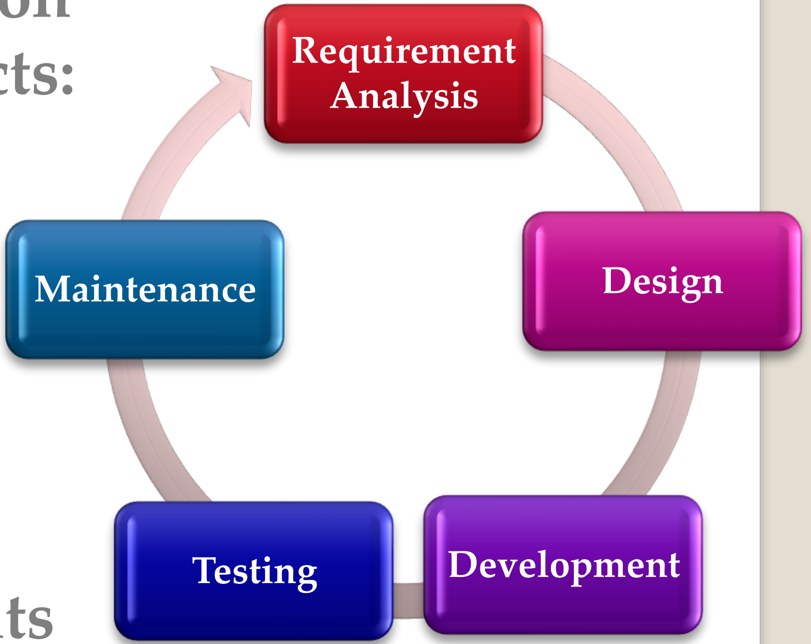


Continued..

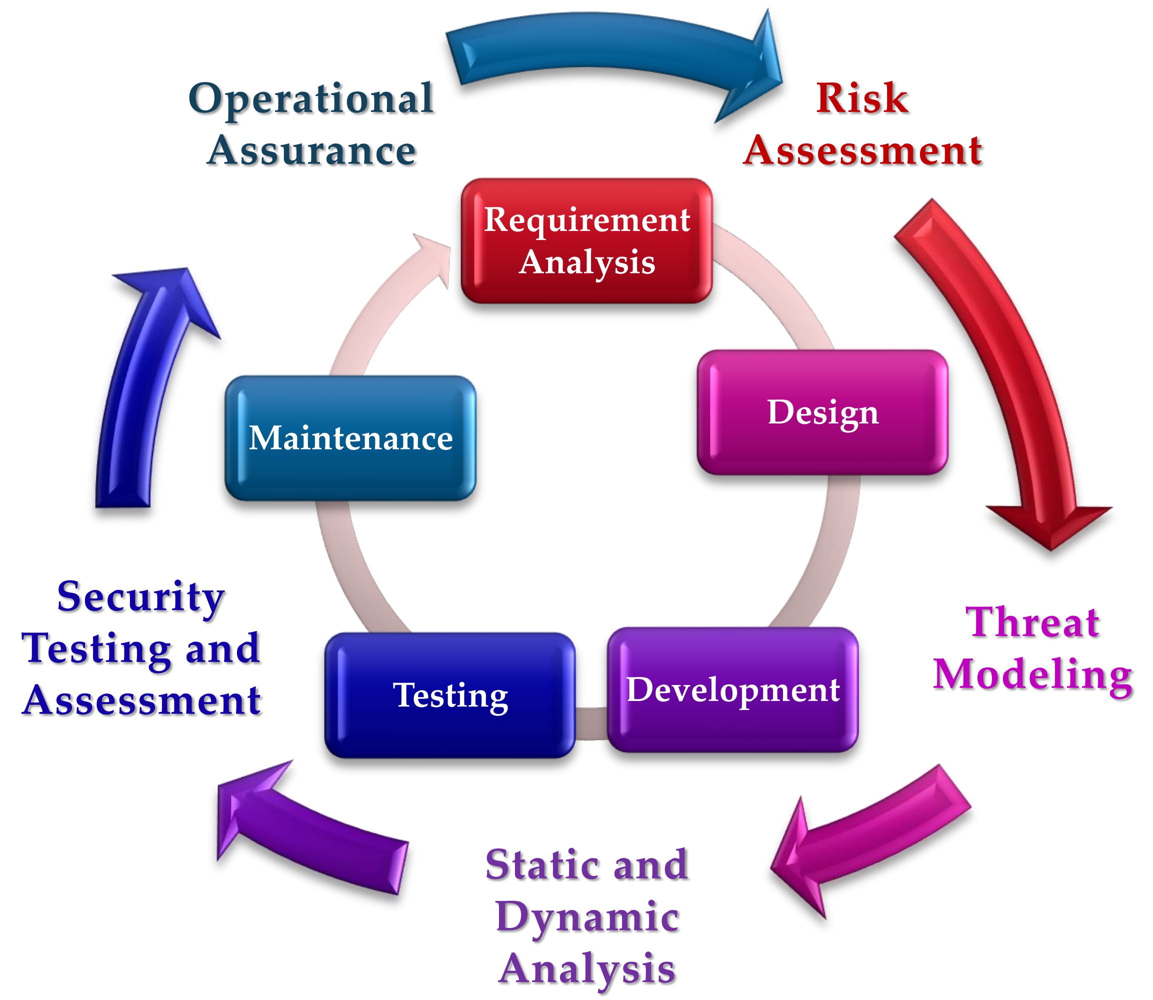
- Software Development Life Cycle
- Secure Software Development Life Cycle
- Threat Modeling
- SAST and DAST tools
- Question and Answers

Software Development Cycle

- A generic methodology can be seen in the foundation of these processes and consist of these crucial aspects:
  - Proposal
  - Production
  - Distribution
  - Maintenance
- Adhering to this strategy is what defines the effectiveness of these life cycle models
- Simplicity and Efficiency Leads to Beneficial Results



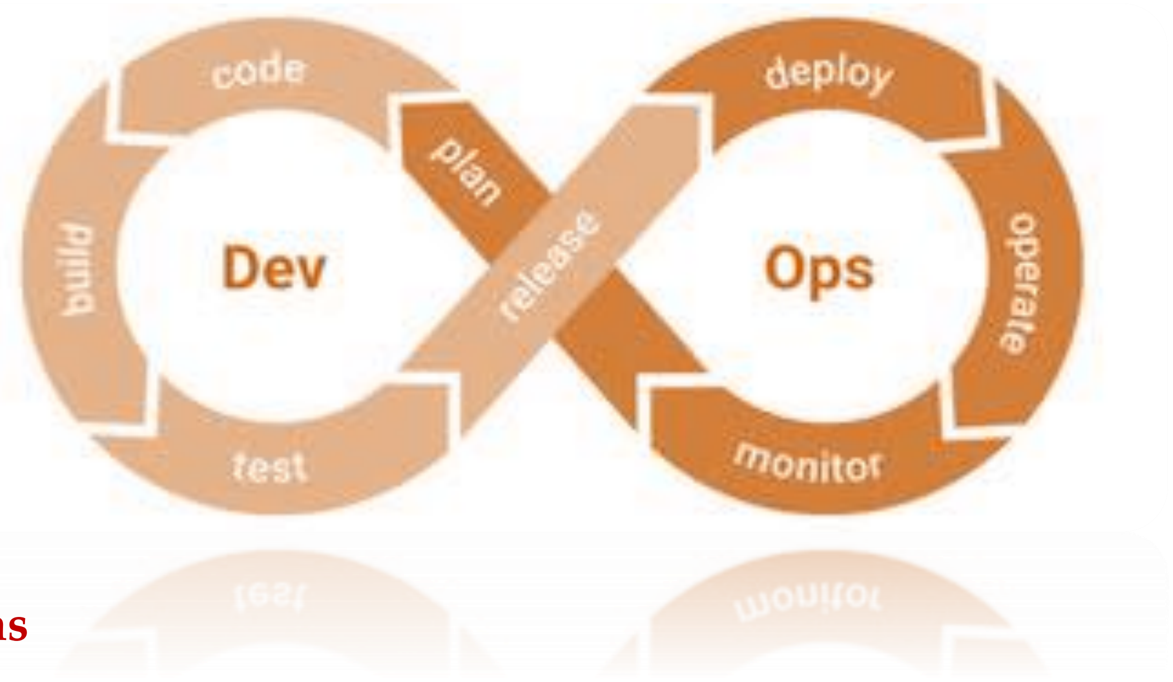
Securing Scientific Software Development



Mitigating the risk of software vulnerabilities with best practices and tools for secure scientific software development

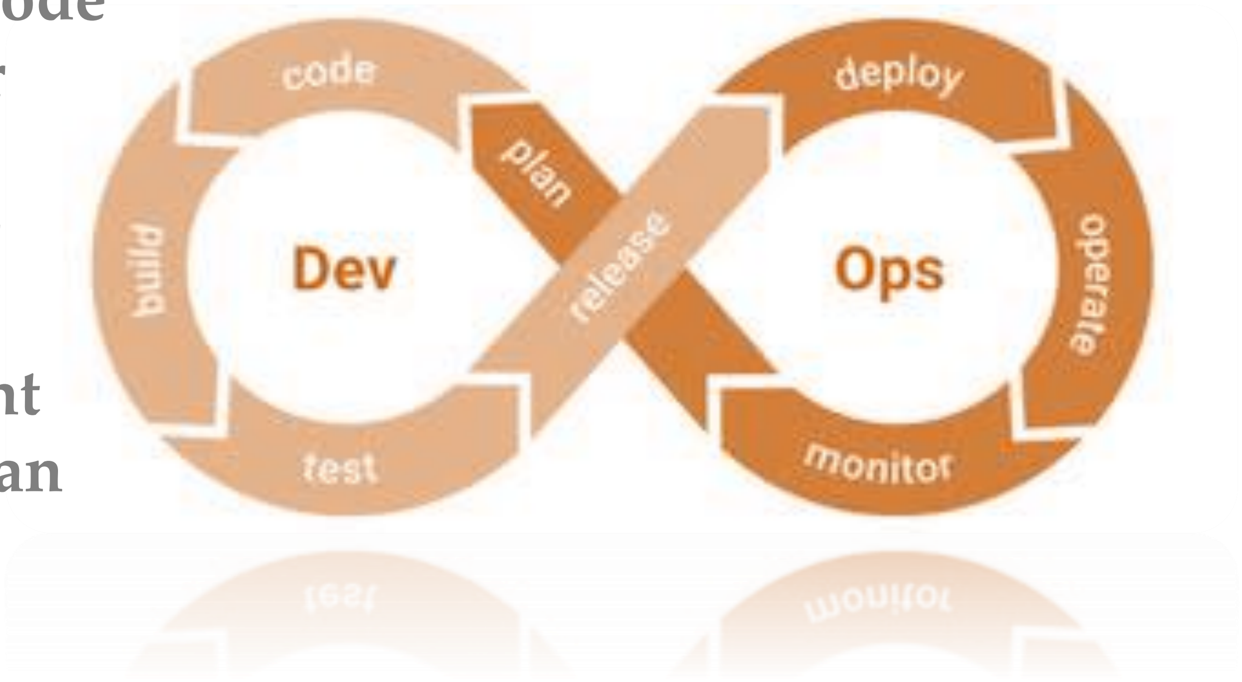
DevOps Life Cycle

- developer team and the IT operations team working together: DevOps
- collaboration helps aids in superior quality throughout the system
- continuous development and continuous integration (CI/CD)
  - continuous testing, continuous deployment, monitoring, feedback, and operations
  - developers can ensure an application's operations and security



DevOps Life Cycle

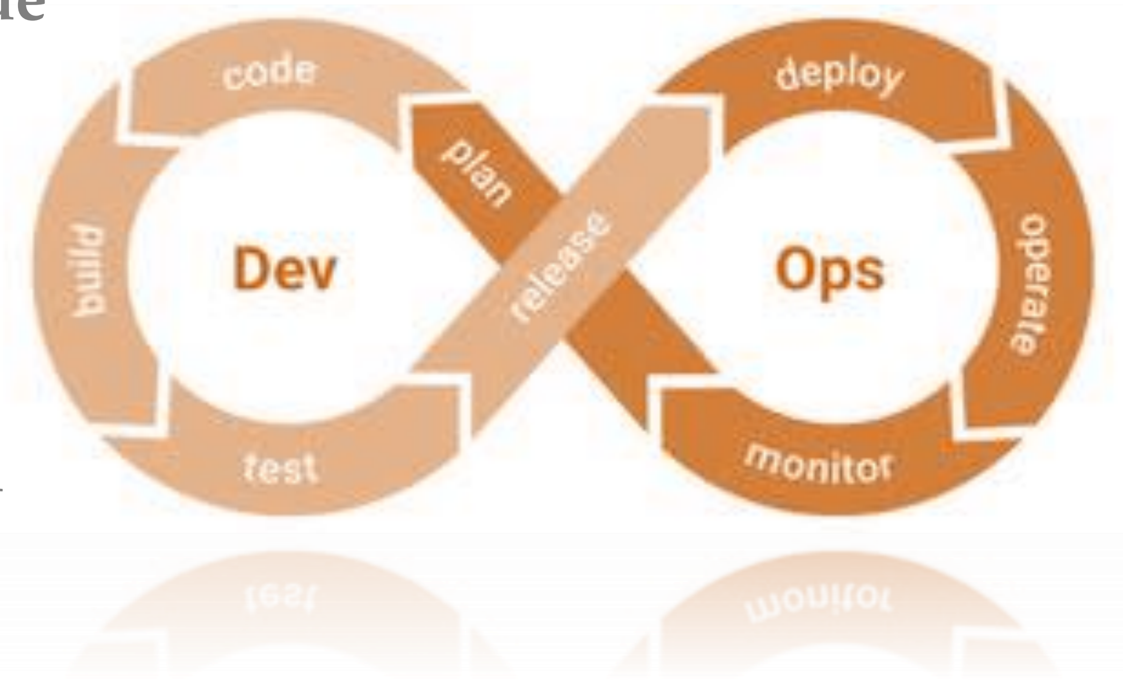
- testing oversees the manufactured code against the problems that arise after compilation.
- utilization of automation tools after developer check
- continuous deployment and efficient risk assessment where developers can take advantage of the running application.



DevOps Life Cycle

- testing oversees the manufactured code against the problems that arise after compilation.
- utilization of au ls after developer ch
- continuous nt risk assess an take advanta application.

The security issues are usually not detected until the software has passed all the tests through the security teams.



Continued..

Secure Software Development Life Cycle

Threat Modeling

SAST and DAST tools

Question and Answers

DevSecOps Life Cycle

- **Plan**

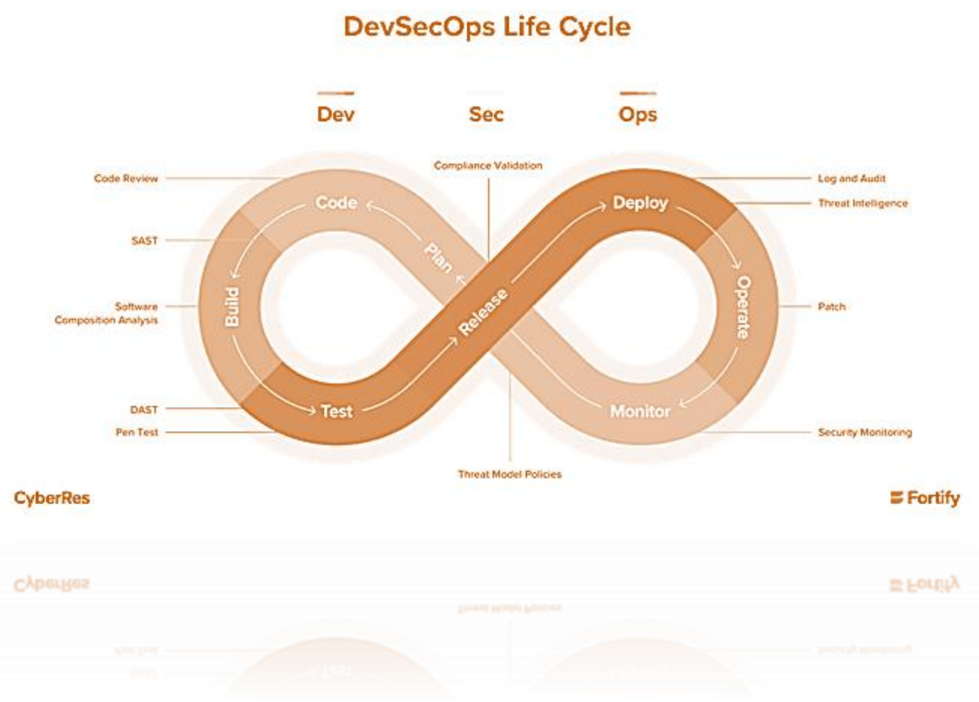
  Addressing technical security such as, investigating firewalls, antivirus software, password management, backups, choosing or creating security policies for production, training staff on the security tools to be used throughout the lifecycle

- **Code**

  Employ IDE Security Plug-ins - each IDE supports a different suite of security plugins. For Visual Studio, security reviewer process all languages and file extensions to investigate and detect hidden weaknesses , highlighting vulnerabilities using Open Web Application Security Project (OWASP), Payment Card Industry Data Security Standard(PCI-DSS), Web Application Security Consortium (WASC), Common Vulnerabilities and Exposures (CVE) or more

- **Build**

  Using Static Application Security Testing (SAST) tools to scan an application's source, binary, or byte code; aids in remediating underlying security flaws. Dynamic Application Security Testing (DAST) used to perform tests the application at runtime, analyzing the web application through front-end vulnerabilities.



DevSecOps Life Cycle

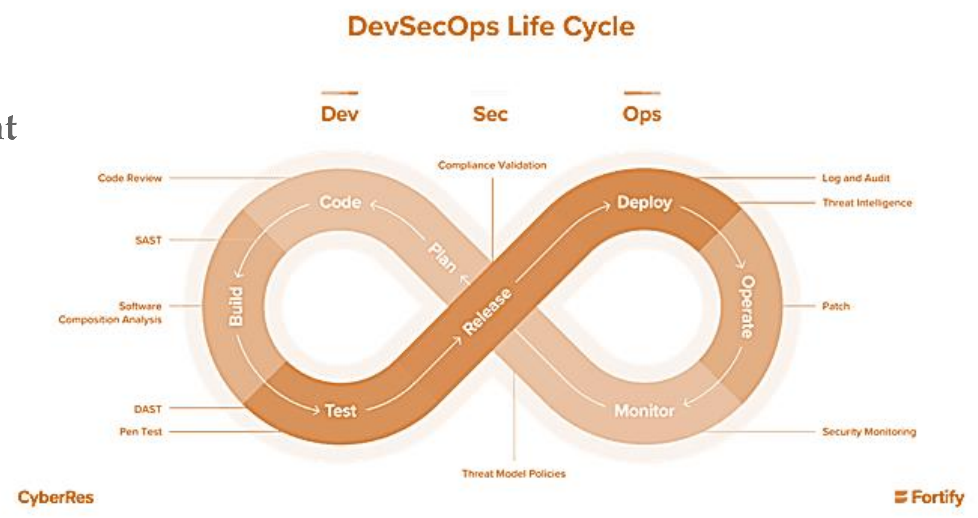
- **Test**
  Chaos Testing is testing production applications by performing health checks and cleaning up unused system resources. Input Fuzzing tests an application by providing invalid, unexpected, and random inputs to the computer

- **Release**
  Continuous Code Signing - A security function that is responsible for defining and implementing corporate security policy as it relates to software development

- **Deploy**
  Signature verification is verifying the integrity of the application's signature to ensure that the application came from the correct developers

- **Operate**
  Runtime Application Self-Protection (RASP) is a security technology that uses runtime instrumentation to detect and block computer attacks by utilizing information from the inside of the running software

- **Monitor**
  User Entity and Behavior Analytics (UEBA) is a type of cybersecurity solution that discovers threats by identifying activity that differs from the baseline behavior



Continued..

Threat Modeling

SAST and DAST tools

Question and Answers

Designing for Security: Threat Modeling

Goals of Threat Modeling

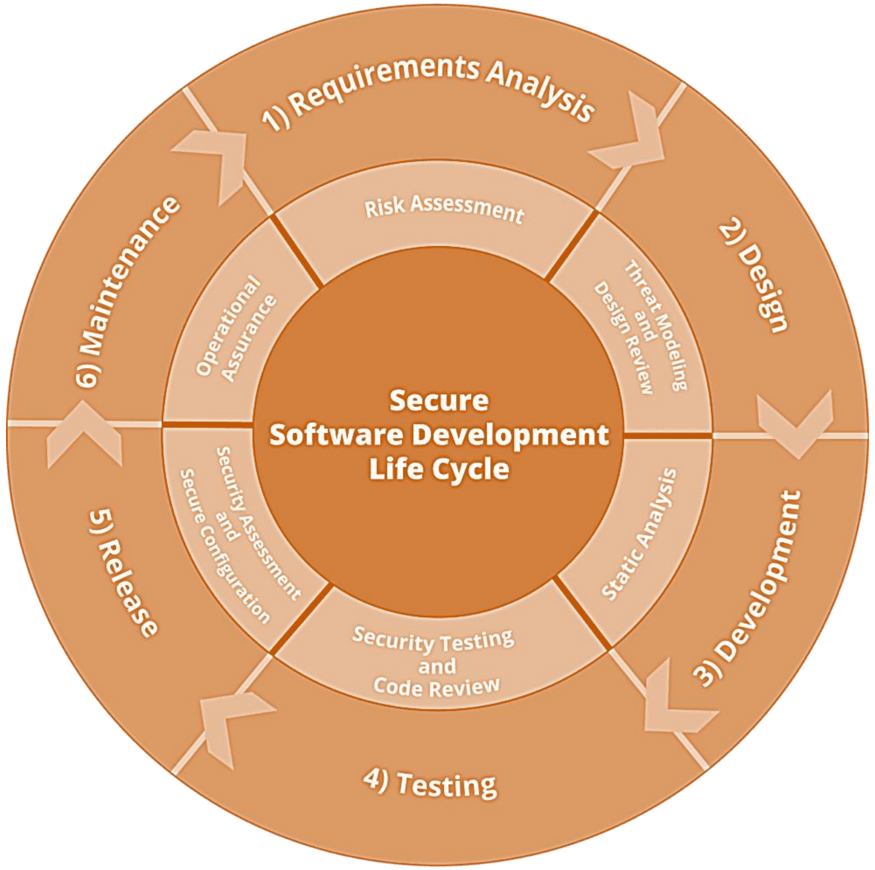
1. Enables organization to anticipate threats rather than reacting to them.
2. Prioritizes resources, allowing for an organization to focus on the most significant vulnerabilities first.
3. Promotes the development of secure software
4. Reduces risk and cost of a cybersecurity incident

What is Threat Modeling?

- Assessing security risks of a software system from an adversary’s perspective

Risk:

- **The potential for loss, damage or destruction of an asset as a result of a threat exploiting a vulnerability**



1) Requirements Analysis
2) Design
3) Development
4) Testing
5) Release
6) Maintenance
Risk Assessment
Threat Modeling and Design Review
Static Analysis
Security Testing and Code Review
Security Assessment and Secure Configuration
Operational Assurance
Secure Software Development Life Cycle

What is Threat Modeling?

- A proactive approach to identifying, managing, and mitigating potential threats.
- Includes defining system components, identifying entry points, recognizing potential threats, categorizing threats, and implementing countermeasures.
- The ultimate goal is improving system security and minimizing cybersecurity risk.

Components of a Threat Model

- System Overview - An understanding of how the system and software function, including how it interacts with other systems
- Assets - Information that needs protected, including any personal data, system configurations, or intellectual property
- Adversaries - Who might be interested in compromising the system.

Components of a Threat Model (cont.)

- Attack Vectors - How Adversaries might attack the system
- Weaknesses & Vulnerabilities - Points where the adversary could exploit the system.
- Mitigations - What measures can be taken to lessen the risk of a vulnerability.

Threat Modeling Process

- System Decomposition - Breaks down the system into assets, users, entry points, and data flows.
- Threat Identification - Identify potential threats from all perspectives.
- Vulnerability Analysis - Identify weaknesses that could be exploited.

Threat Modeling Process

- Risk Assessment - Estimate the impact and likelihood of each threat and vulnerability
- Mitigation Strategy - Develop strategies to reduce the risks
- Document & Communicate - Keep a record of all findings, actions, and unresolved risks
- Review & Update - Update the model as threats evolve.

STRIDE Methodology

STRIDE Methodology Overview

- Developed by Microsoft
- Used to identify and categorize potential threats
- Typically used during the design phase of a system
- Covers mainly technical aspects

STRIDE Threat Categories

- Spoofing Identity
- Tampering with Data
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege

Spoofing Identity

- Definition - An attacker impersonates another user.
- Types of Spoofing - Identity, IP, ARP, and DNS spoofing
- Potential Damages -Unauthorized access, stolen data, damaged reputation
- Mitigation Techniques - Two-factor authentication, encryption, and education

Tampering with Data

- Definition - Unauthorized alteration of data
- Types of Tampering - Data, code, or configuration tampering
- Potential Damages - Unauthorized access, false information, and lack of data integrity
- Mitigation Techniques - Checks for Data integrity, secure transmission protocols, and restrictions on access

Repudiation

- Definition - A user denies having performed an action
- Types of Repudiation - Transaction, email, and contract repudiation
- Potential Damages - Business disputes, auditing problems, and inability to enforce accountability
- Mitigation Techniques - Digital signatures, authentication protocols, and monitoring

Information Disclosure

- Definition - Unauthorized access to sensitive information
- Types of Information Disclosure - Data leaks and breaches
- Potential Damages - Damaged reputation, regulatory penalties, financial loss, and a loss of trust
- Mitigation Techniques - Access controls, encryption, data masking, and education

Denial of Service

- Definition - Making a system unavailable to users
- Types of DoS - Network, application, and system level attacks
- Potential Damages - Financial damage and loss of trust
- Mitigation Techniques - Firewalls, capacity planning, and traffic filtering

Elevation of Privilege

- Definition - A user gaining higher access privileges than intended.
- Types of Elevation of Privilege - Role, Access, and privilege escalation
- Potential Damages - System manipulation, system damage, and unauthorized access to sensitive data
- Mitigation Techniques - Access controls, patch management, regular auditing, and principle of least privilege

Limitations of STRIDE

- No built-in method for risk scoring
- Focuses mainly on technical threats, which leaves out physical security threats
- Relies heavily upon the created data flow diagram
- Identifying threats and using the model requires a high level of expertise.

DREAD Methodology

DREAD Methodology Overview

- Introduced by Microsoft
- Designed to evaluate and assess the risk of threats
- Often used with other methodologies, like STRIDE
- Aims to prioritize resources, addressing the most significant threats first.

DREAD Methodology

- Damage Potential
- Reproducibility
- Exploitability
- Affected Users
- Discoverability

Damage Potential

- Definition - How bad an attack is
- Types of Damage Potential - Data loss, service interruption, financial damage, and reputation damage
- Consequences - Dependent on the severity of the attack
- Mitigation Techniques - Regular backups, incident response planning, and disaster recovery planning

Reproducibility

- Definition - How easy is an attack is reproduced
- Types of Attacks - Exploit scripts, automated attacks, and manual attacks
- Consequences- Higher reproducibility results in more frequent damage
- Mitigation Techniques - Patching vulnerabilities and implementing intrusion detection systems

Exploitability

- Definition - What is needed to launch an attack
- Types of Exploitability - No user interaction to extensive user interaction
- Consequences - The easier it is to exploit an attack, the higher the risk of one occurring.
- Mitigation Techniques - Education, secure programming practices, and regular vulnerability scanning

Affected Users

- Definition - How many users are affected
- Types of Affected Users - single users, groups of users, or all users
- Consequences - The more users affected, the more damage that will occur
- Mitigation Techniques - Principle of least privilege, network segmentation, and access controls

Discoverability

- Definition - How easy the threat is to discover ?
- Types of Discoverability - Threats that are easy to discover, to threats that are extremely difficult to discover
- Consequences - The more likely a threat is to be discovered, the more likely it is to be exploited
- Mitigation Techniques - Regular penetration testing, vulnerability scanning, and security audits

STRIDE & DREAD Risk Assessment

- STRIDE and DREAD are often used together, with STRIDE being used for threat modeling and DREAD being used for risk assessment.

1. STRIDE identifies and categorizes all risks based on the data flow diagram.
2. DREAD assesses the risks and gives them a score.
3. Prioritize threats based on the score

STRIDE & DREAD Risk Assessment

4. Develop Mitigation strategies for each threat, starting with the threat with the highest DREAD score.

5. Apply the strategies, then reassess the DREAD score until it is up to the organizations standards.

6. Document all strategies and changes.

7. Perform the risk assessment regularly.

STRIDE & DREAD Risk Assessment

4. Develop Mitigation strategies for each threat, starting with the threat with the highest DREAD score.

5. Apply the strategies, then reassess the DREAD score until it is up to the organizations standards.

6. Document all strategies and changes.

7. Perform the risk assessment regularly.

Using STRIDE

- Understanding the Adversary’s View
  - Identify all valuable assets and characterize system security using use cases and misuse cases
  - Create data flow diagrams
- Identify threats, using STRIDE’s categories
- Determine risk level
- Develop mitigation strategies
- Document all threats, risks, and mitigation strategies

Using STRIDE Example

- Car optimizer:

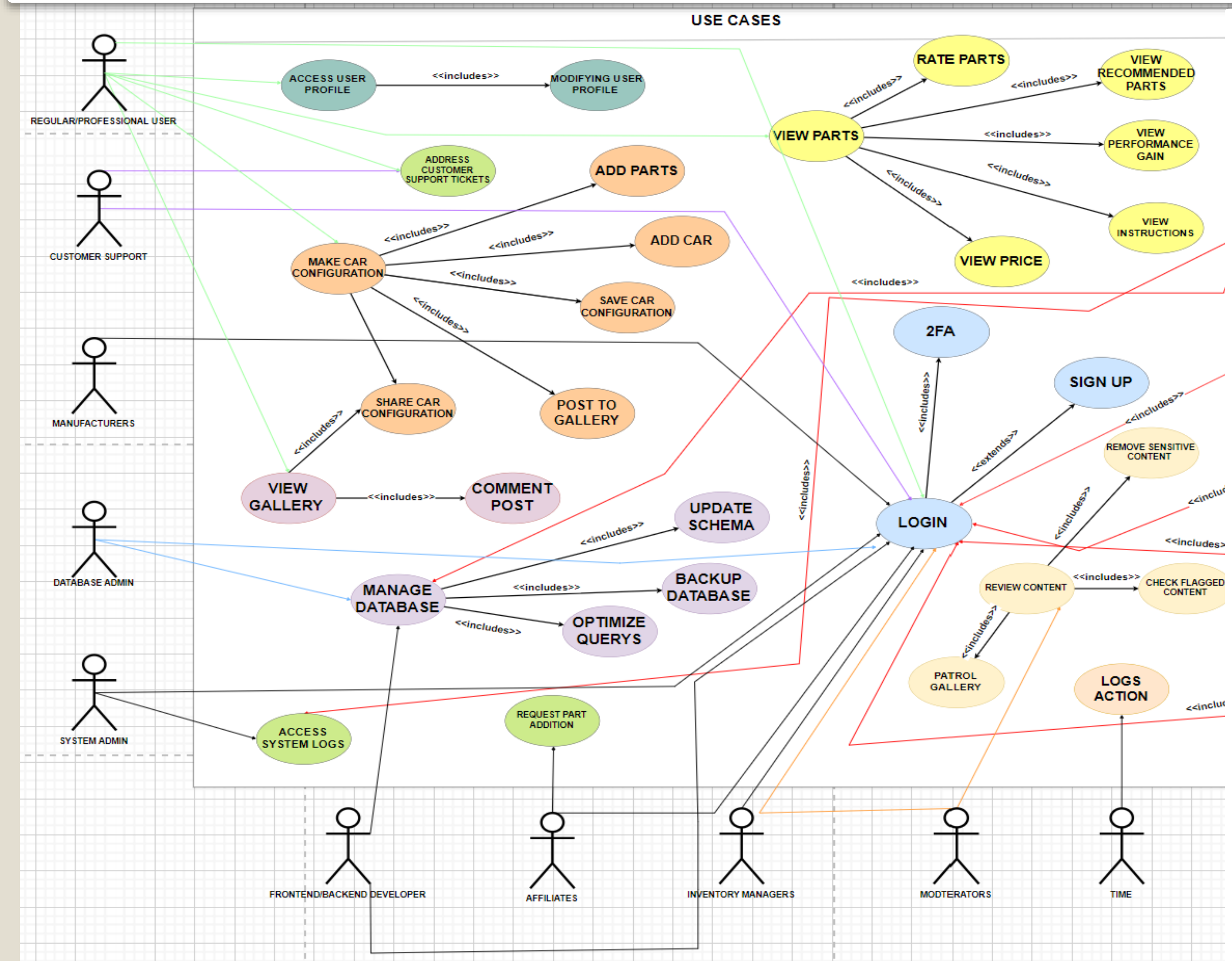
The software system is used for individuals who want to optimize their cars' performance. The software will feature numerous makes and models of vehicles for the user to choose from, covering all the main manufacturers. After they choose their car, they will be able to view its performance from the factory. In addition to the car's factory performance, they will be able to choose different parts for their car and see how their car will perform with the performance enhancements installed. Not only will the user be able to see the performance of their vehicle with these new parts installed, but they will also be able to see the cost and a guide on installation. Users will be able to configure different vehicle specifications and see how they will affect performance compared to how the car comes from the factory.

Use Cases (Software Requirements)

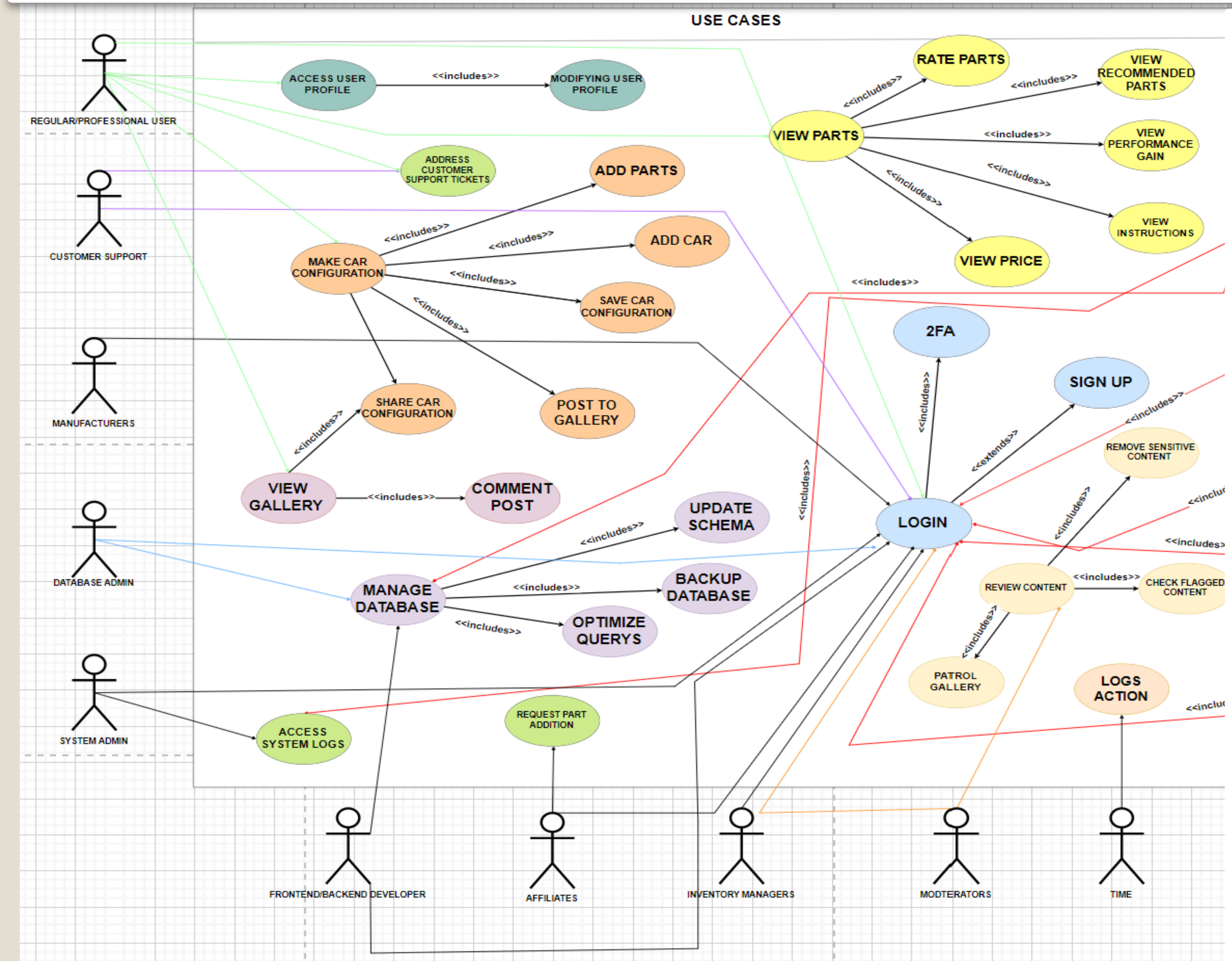
- USE CASES

1. Login: Includes Authentication(2FA), Mitigate Brute Force Logins, Extends Sign Up
2. View Parts: Includes view recommended parts, view price, view instructions, rate parts, view performance gain
3. Make Car Configuration: Includes add car, share car configuration, post to gallery, save car config, add parts
4. Manage Database: Includes backup database, optimize queries, update schema
5. View Gallery: Includes comment post, share car configuration
6. Access User Profile: Includes Modifying User Profile
7. Access System Logs
8. Logs Action (Done by time subject)
9. Manage User Access Controls
10. Address Customer Support Tickets (Customer Support)
11. Request Part Addition (Affiliates)
12. Review Content: Includes check flagged content, remove sensitive content, patrol gallery (Moderator)

Use Case Scenarios



Use Case Scenarios



Missing Security Requirements

Misuse Cases (Addressing Security Attacks)

- TAMPER DATABASE
- Attack: Attackers may try to tamper with the database or system logs to manipulate them.
- Mediation: Checking access controls to ensure that unauthorized access cannot be gained. We also will implement two factor authentication to further prevent this.

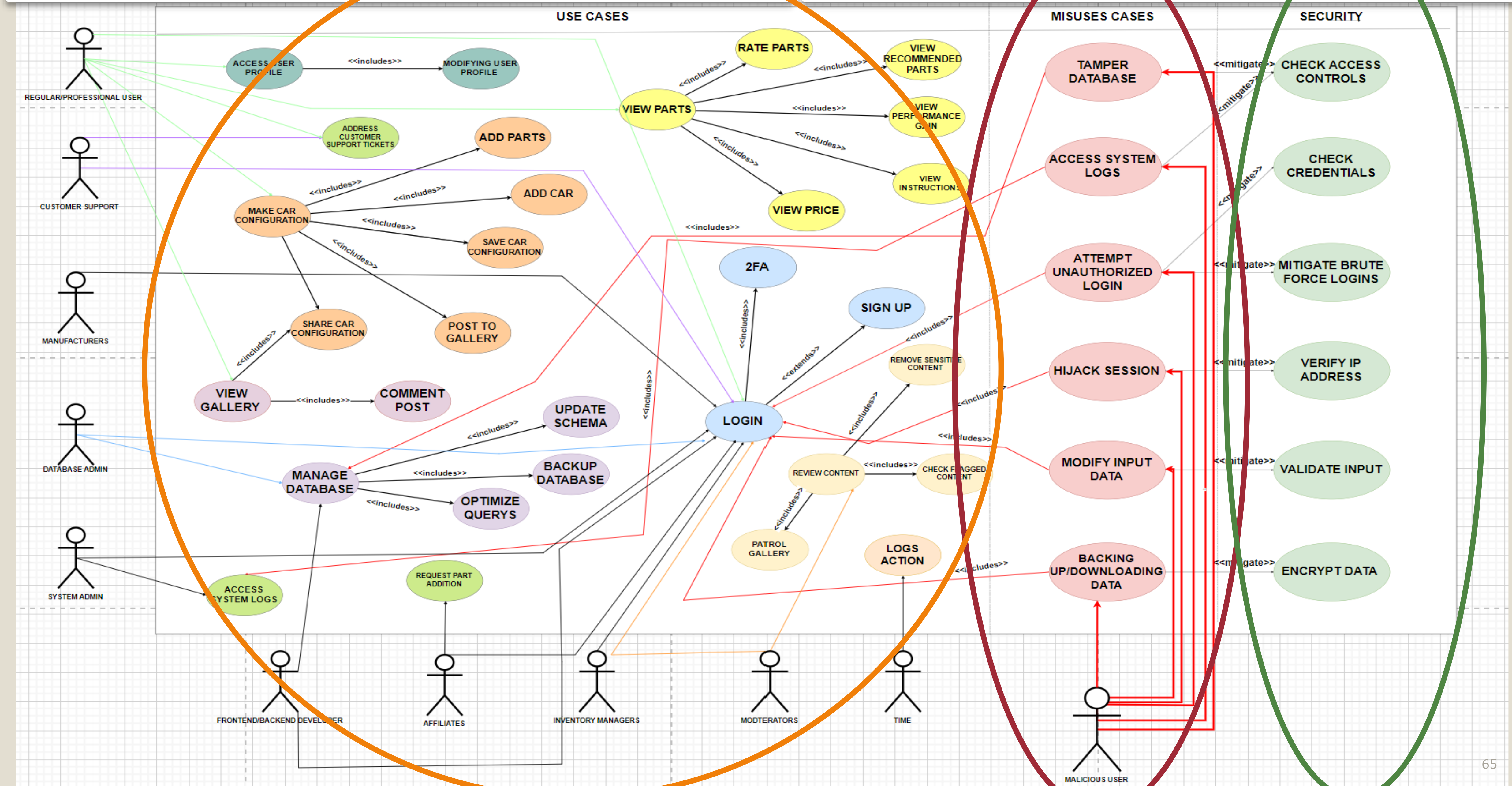
- UNAUTHORIZED LOGIN
- Attack: Attempting to gain access that they are not authorized to have
- Mediation: Checking credentials, limiting multiple login attempts, two-factor authentication

- MODIFY INPUT DATA
- Attack: Attempting to input faulty or malicious data into the database through the frontend
- Mediation: Sanitize and validate user input

- STEAL USER INFORMATION
- Attack: Attempting to view user database tables to gain access to sensitive information.
- Mediation: Access controls, encrypting stored sensitive data

- ACCESS SYSTEM LOGS
- Attack: Access and manipulate system logs
- Mediation: Access controls, two-factor authentication

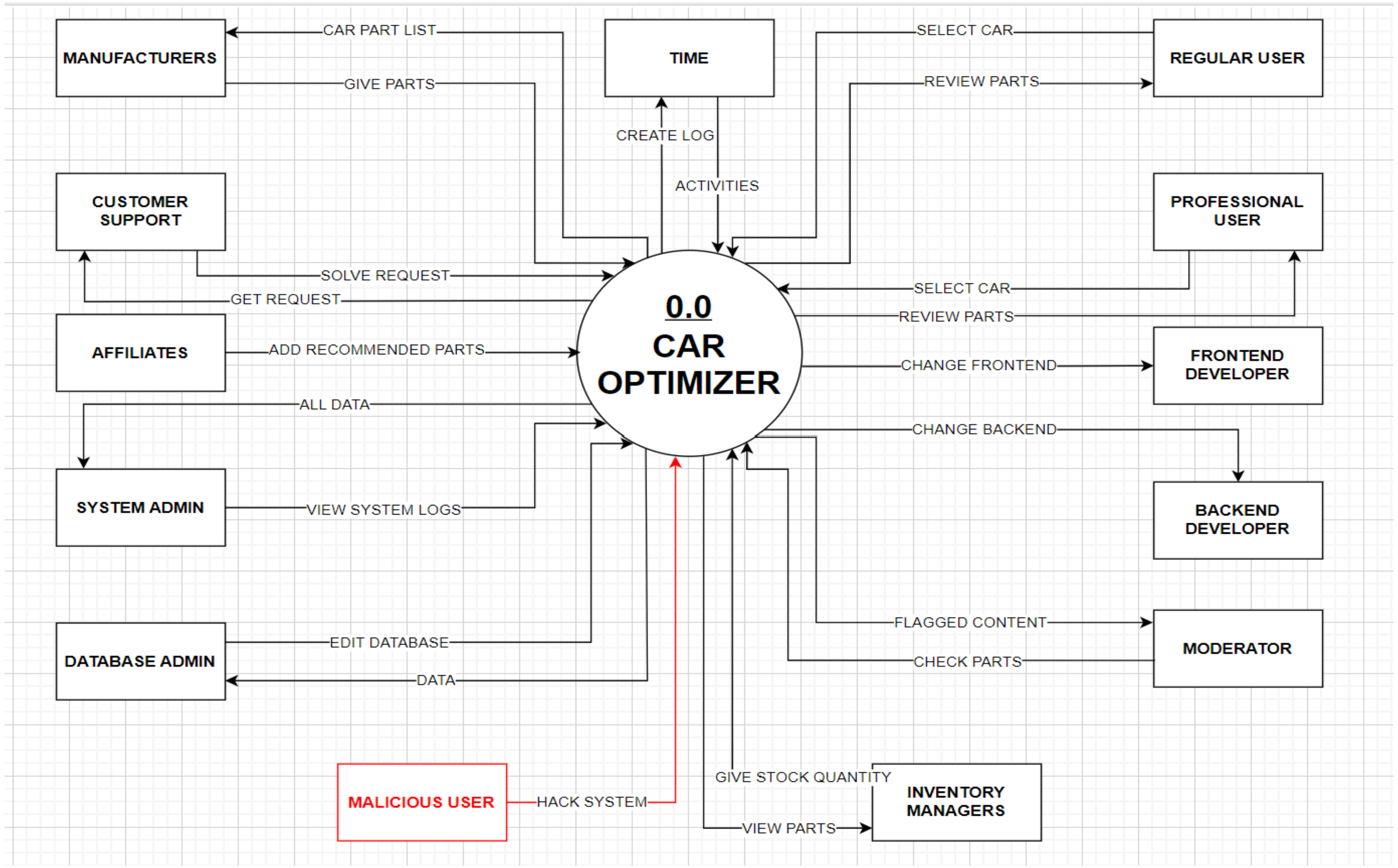
Use Case Scenarios Misuse Scenarios



Creating Data Flow Diagrams (Context level)

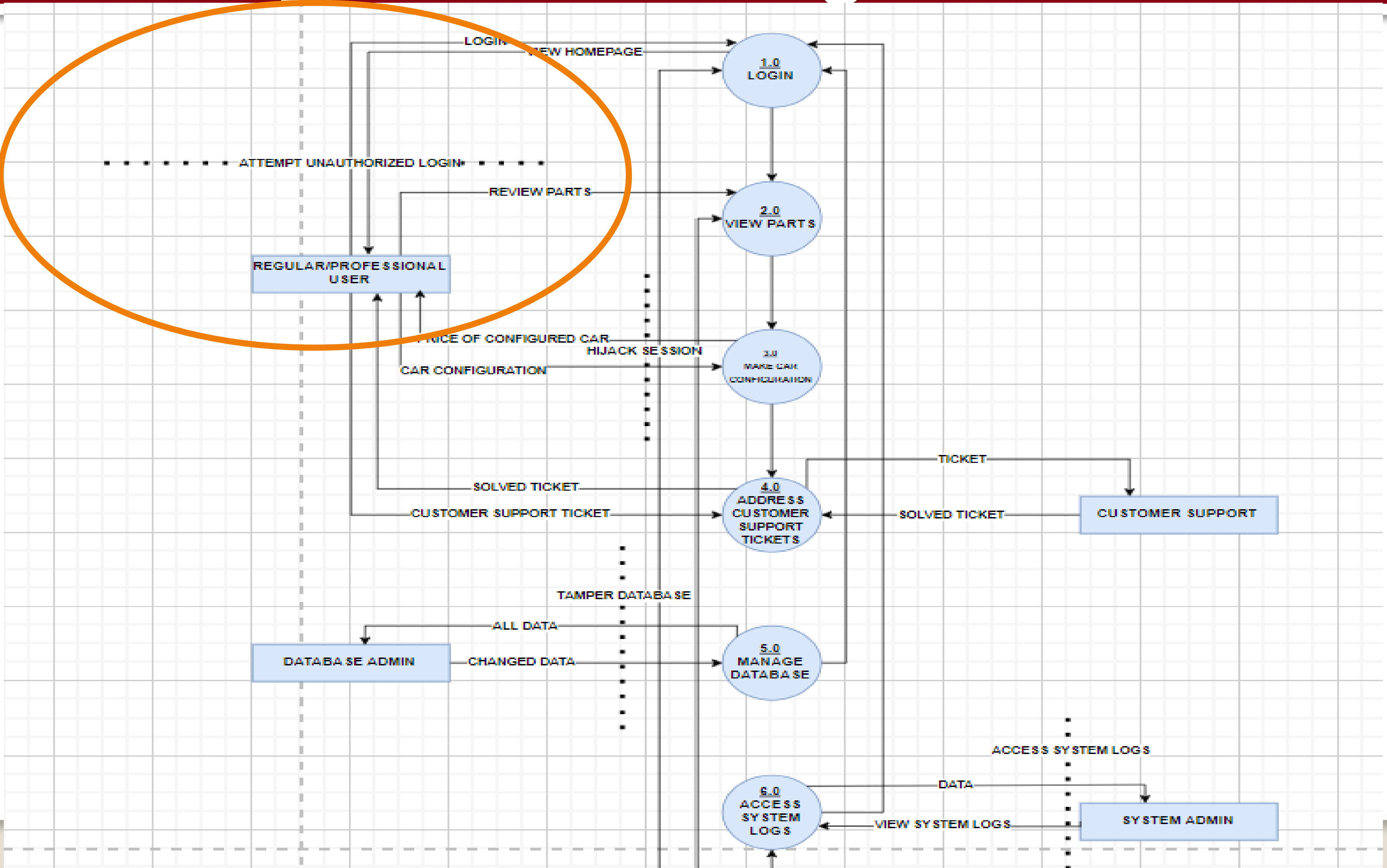
- Visual model of how system processes data and what are the entry points

•

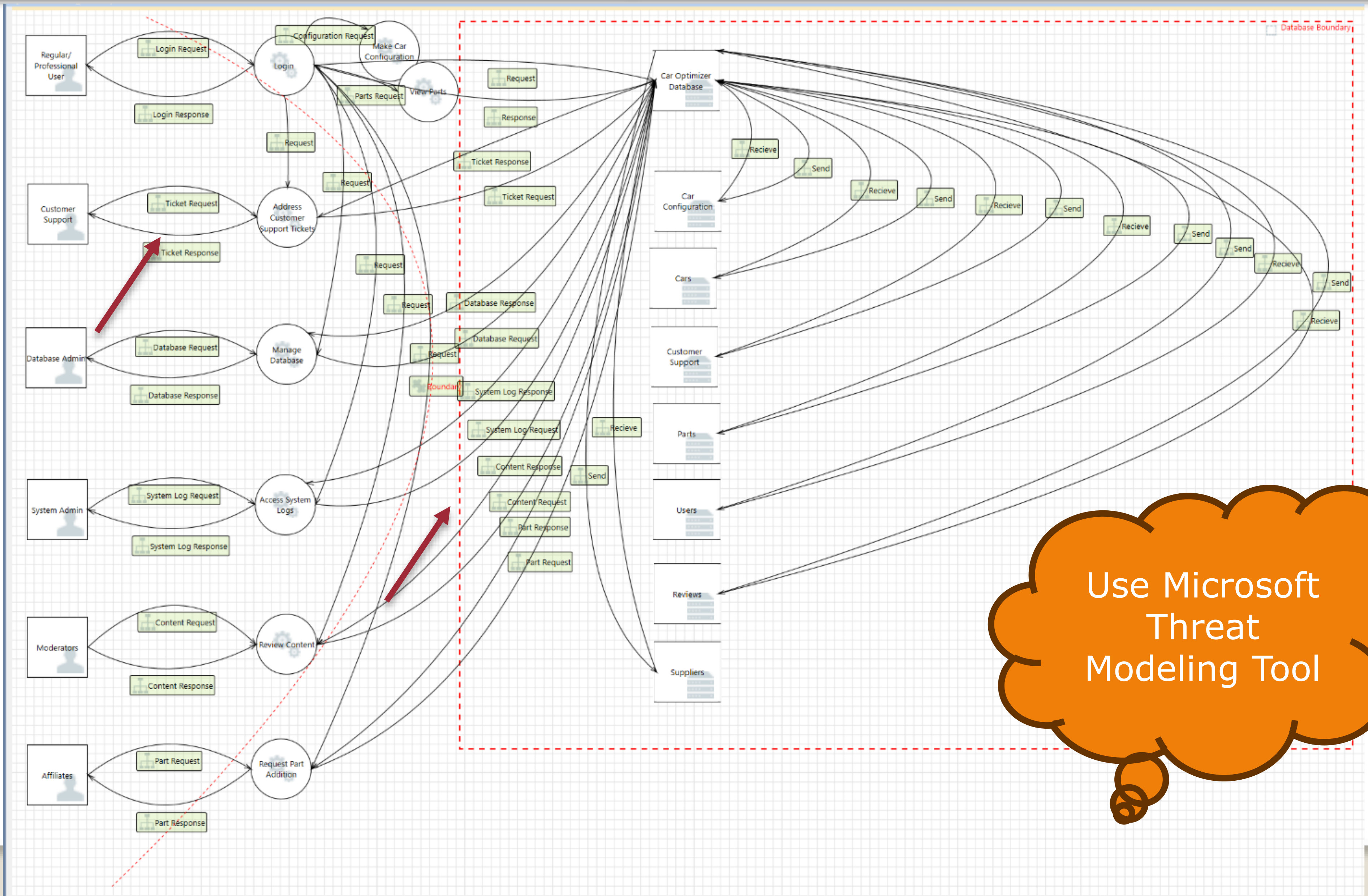


Creating Data Flow Diagrams (Level 0)

•



Identify Threats



Use Microsoft Threat Modeling Tool

Analyze Threats using STRIDE

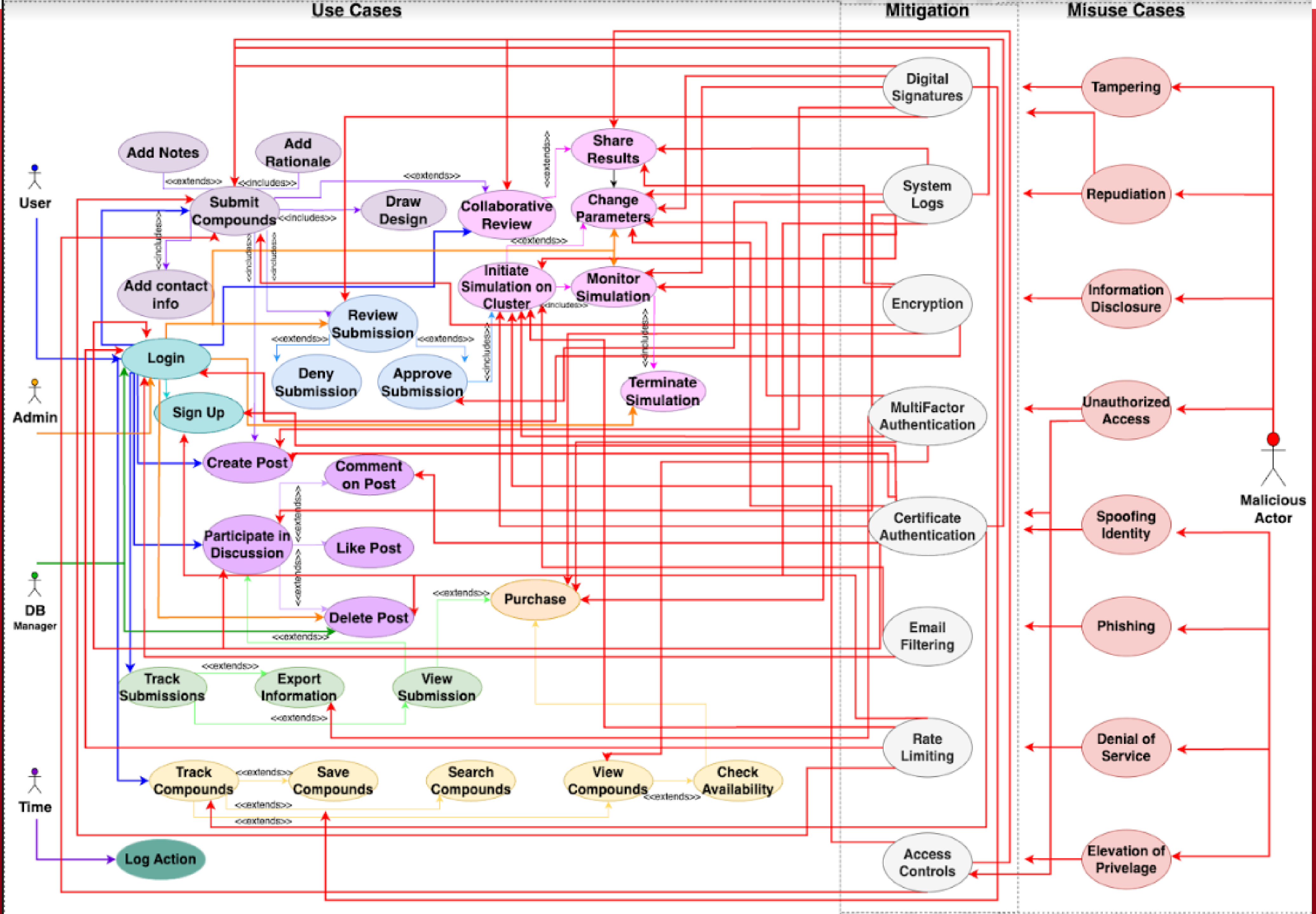
Potential Process Crash or Stop for Review Content [State: Not Started] [Priority: High]

Category:	Denial Of Service
Description:	Review Content crashes, halts, stops or runs slowly; in all cases violating an availability metric.
Justification:	Thorough code reviews, debugging, and the implementation of failover mechanisms to maintain system availability in the event of a process crash. Monitoring system performance and promptly addressing issues through regular maintenance can prevent prolonged service disruptions.

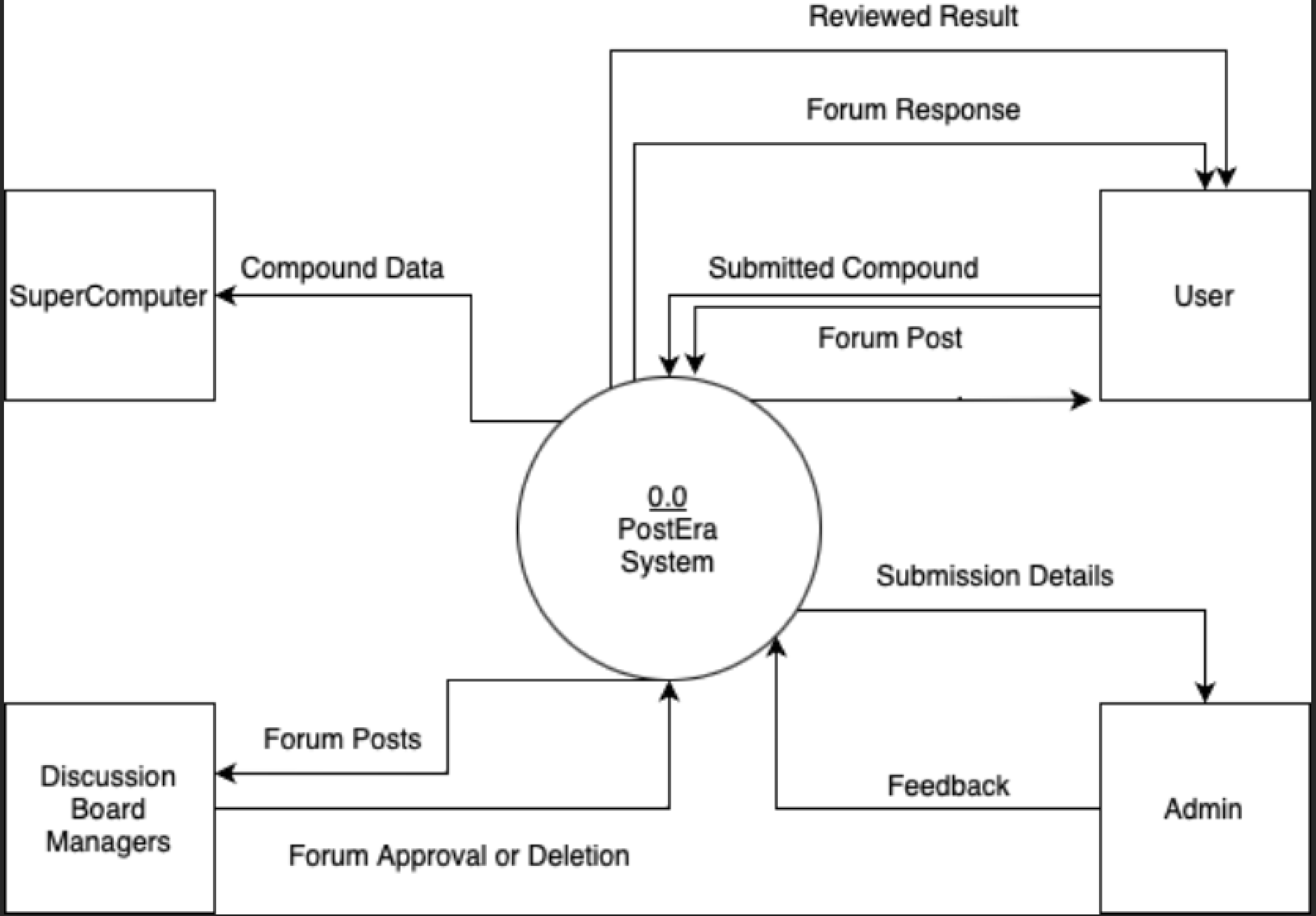
Using STRIDE Example

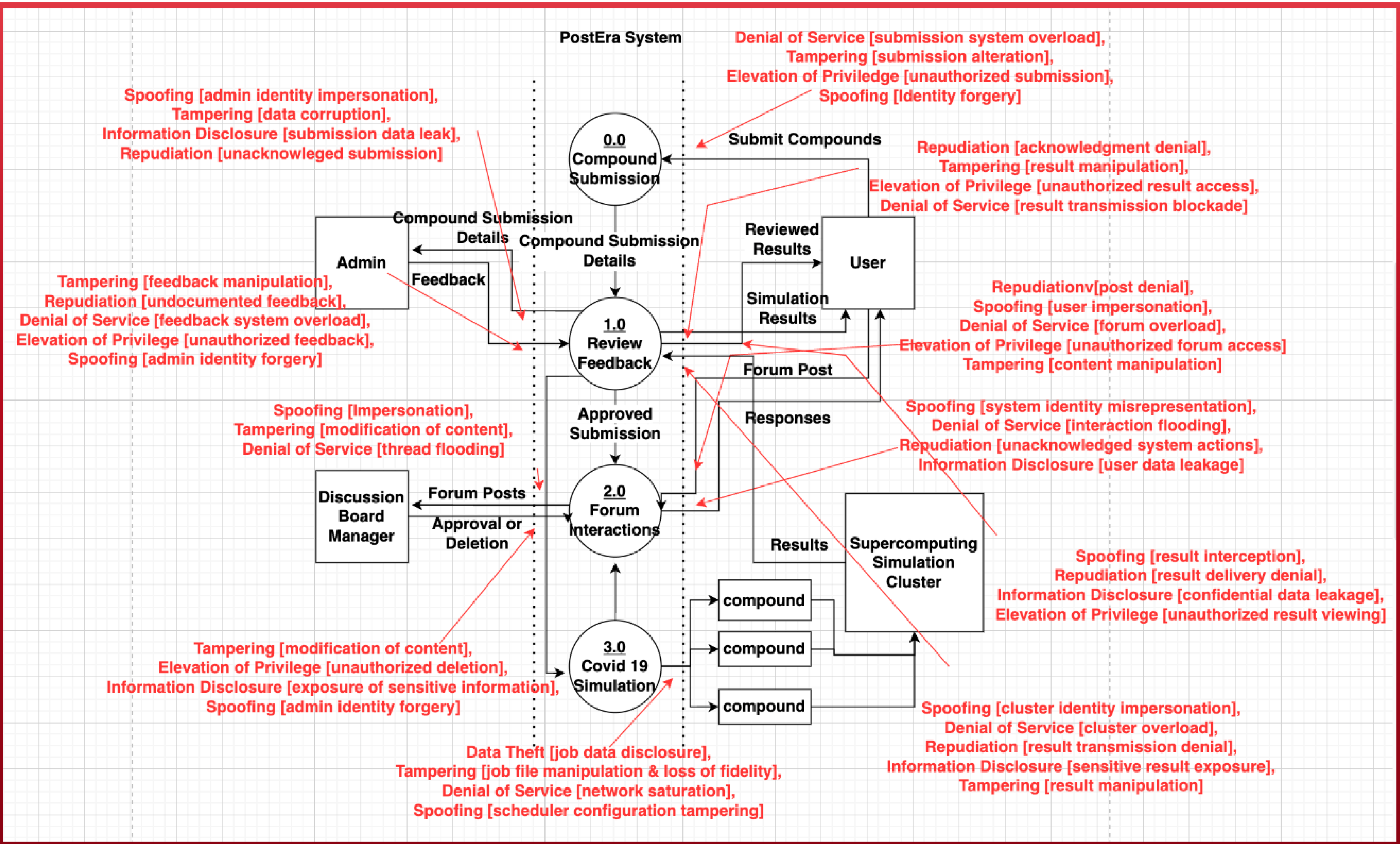
▶ **PostEra.ai:**

PostEra.ai is a platform designed for the collaborative development of therapeutic compounds against COVID-19. It allows researchers and scientists to contribute their compound designs by submitting molecular structures. The site facilitates the collection, analysis, and prioritization of these compounds for synthesis, simulation and testing, providing a unique collaborative approach to accelerate COVID-19 drug discovery.


Use Cases
Mitigation
Misuse Cases
User
Admin
DB Manager
Time
Malicious Actor
Add Notes
Add Rationale
Submit Compounds
Draw Design
Collaborative Review
Share Results
Change Parameters
Add contact info
Initiate Simulation on Cluster
Monitor Simulation
Terminate Simulation
Review Submission
Deny Submission
Approve Submission
Login
Sign Up
Create Post
Comment on Post
Participate in Discussion
Like Post
Delete Post
Purchase
Track Submissions
Export Information
View Submission
Track Compounds
Save Compounds
Search Compounds
View Compounds
Check Availability
Log Action
Digital Signatures
System Logs
Encryption
MultiFactor Authentication
Certificate Authentication
Email Filtering
Rate Limiting
Access Controls
Tampering
Repudiation
Information Disclosure
Unauthorized Access
Spoofing Identity
Phishing
Denial of Service
Elevation of Privelage
<<extends>>
<<includes>>

Context Level Data Flow Diagram




PostEra System
0.0 Compound Submission
1.0 Review Feedback
2.0 Forum Interactions
3.0 Covid 19 Simulation
Admin
User
Discussion Board Manager
Supercomputing Simulation Cluster
compound
compound
compound
Submit Compounds
Compound Submission Details
Compound Submission Details
Feedback
Reviewed Results
Simulation Results
Forum Post
Responses
Approved Submission
Forum Posts
Approval or Deletion
Results
Denial of Service [submission system overload],
Tampering [submission alteration],
Elevation of Priviledge [unauthorized submission],
Spoofing [Identity forgery]
Spoofing [admin identity impersonation],
Tampering [data corruption],
Information Disclosure [submission data leak],
Repudiation [unacknowleged submission]
Repudiation [acknowledgment denial],
Tampering [result manipulation],
Elevation of Privilege [unauthorized result access],
Denial of Service [result transmission blockade]
Tampering [feedback manipulation],
Repudiation [undocumented feedback],
Denial of Service [feedback system overload],
Elevation of Privilege [unauthorized feedback],
Spoofing [admin identity forgery]
Repudiationv[post denial],
Spoofing [user impersonation],
Denial of Service [forum overload],
Elevation of Privilege [unauthorized forum access]
Tampering [content manipulation]
Spoofing [Impersonation],
Tampering [modification of content],
Denial of Service [thread flooding]
Spoofing [system identity misrepresentation],
Denial of Service [interaction flooding],
Repudiation [unacknowledged system actions],
Information Disclosure [user data leakage]
Spoofing [result interception],
Repudiation [result delivery denial],
Information Disclosure [confidential data leakage],
Elevation of Privilege [unauthorized result viewing]
Tampering [modification of content],
Elevation of Privilege [unauthorized deletion],
Information Disclosure [exposure of sensitive information],
Spoofing [admin identity forgery]
Spoofing [cluster identity impersonation],
Denial of Service [cluster overload],
Repudiation [result transmission denial],
Information Disclosure [sensitive result exposure],
Tampering [result manipulation]
Data Theft [job data disclosure],
Tampering [job file manipulation & loss of fidelity],
Denial of Service [network saturation],
Spoofing [scheduler configuration tampering]

Continued..

SAST and DAST tools

Question and

Answers

SAST and DAST Tools

Static Analysis

- may look for generic defects, or focus on "code cleanliness" (maintainability, style, "quality"etc.)
- Some defects are security vulnerabilities
  - Java users: Consider quality scanners SpotBugs (successor of FindBugs) or PMD
  - Cppcheck (C++, works with C)

    Assign average of numbers to node.
  - Attacks Trees:Propagate risk values to parent nodes.
    - Sum risk values if child nodes are ANDed together.
    - Use highest risk value of all children if nodes are ORed together.

Dynamic Analysis

  - OWASP Zed Attack Proxy(ZAP)- Free web security tool
  - Penetration Testing- Combination of both Static and Dynamic Analysis.

What is Static Analysis?

- Static Analysis is a method of computer program debugging that is done by examining the code without executing the program.
- Static analysis process is also useful for addressing weaknesses in source code that could lead to buffer overflow -- a common software vulnerability.
- Static analysis is used in software engineering by software development and quality assurance terms.

What is Static Analysis?

- Static Analysis
  debugging tha
  without execut
- Static analysis
  weaknesses in
  overflow -- a c
- Static analysis
  software devel

- supports a wide variety of static checks which includes:
  - Automatic variable checking
  - Bounds Checking for array overruns
  - Classes checking
  - Memory leaks
  - Resource leaks
  - Dead code elimination
  - Performance errors
  - Undefined variables

Dynamic Analysis

Penetration testing (pen testing)

- Pretend to be adversary, try to break in
- Depends on the skills of the pen testers
- Really a combination of static & dynamic approaches

Tools

- Often convenient to have a pre-created set of tools
- Kali Linux (successor to BackTrack):
- Linux distribution based on (widely-used) Debian
- Preinstalled with over 600 penetration-testing programs, including nmap (a port scanner), Wireshark (a packet analyzer), and OWASP ZAP
- Can run natively when installed on a computer's hard disk, can be booted from a live CD or live USB, or it can run within a virtual machine
- Useful for pen testing applications before release

5 phases of Penetration Testing

- Reconnaissance- trying to discover domain names, gather any set of intelligence
  - Active – using the network
  - Passive – not touching the network
- Scanning- port scanning, network sniffing (looking for open port, service), vulnerability scanner, data gathering
- Gaining access to the applications, system, network (to get access, control to the system)
- Maintaining Access- installing backdoor to maintain access,
- Covering tracks- once we get access, then we need escape the security (clearing cache, browsing history)

Conclusion

SSD:

1. Enables organization to anticipate threats rather than reacting to them.
2. Prioritizes resources, allowing for an organization to focus on the most significant vulnerabilities first.
3. Promotes the development of secure software
4. Reduces risk and cost of a cybersecurity incident

Blog:

https://bssw.io/blog_posts/secure-software-programming-practices-and-development

THANKS,

For questions, please email at nitin.Sukhija@sru.edu