## Exam 1 [Quizlet](https://quizlet.com/class/8600507/)

### Fundamentals of Software Security

1. explain the definition of a security vulnerability

= a bug that results in security consequences

Design flaw OR poor coding => exploit software

1. list and describe the most common security objectives

Confidentiality = “data is disclosed only as intended”

(B cannot access A’s infor)

Integrity = system/components guards against improper modification

(request login --> no modification on the request/response)

Availability = operational and accessible system/components when it is required for use

(Jenkins went down)

Identification/authentication = claimed identity is valid

Accountability = actions can be traced to actor

Privacy = actor can understand and control their information

1. explain how a vulnerability affects common security objectives

It is a weakness that allows the attacker to compromise the security objectives

1. classify a vulnerability as a design flaw or implementation bug

Design flaw = mistake/weakness in the way software is designed

Eg. no password needed to login

Implementation bug = error/fault in a piece of software

Eg. script allowed when entering password

1. describe common misconceptions about software security
   1. Software security can be accomplished with single tool : the best tools may identify 50% security vulnerabilities
   2. Modern programming language will ensure my software is secure: programming languages were not designed with security as the primary goal. Misuse build-in functionality of programming languages
   3. Only software in sensitive domains need to be secured : more sensitive data included in the future
   4. My software is not important enough to be targeted : any type of data can be important for future attack
   5. Firewall is enough to secure software : design flaw
   6. Cryptography : design flaw
   7. Finding bugs is the goal of software security : if you don’t fix the bug; design flaw
   8. Security is only the developer’s problem : manager, organization’s problem
2. formulate a response to people who state common software security misconceptions

See 5

1. explain why a single tool or practice does not guarantee security of software

the best tools may identify 50% security vulnerabilities

1. describe the three pillars of software security

Risk management to identify/rank/track/understand the risks throughout software development lifecycle

Touchpoints = best practices that software teams use at specific phases of software development

Knowledge = gathering/sharing security knowledge as a foundation for software security practices

1. list the seven touchpoints/best practices for developing secure software(code -> archi penetration -> risk & abuse -> note)

Code review, architectural risk analysis, penetration testing, risk-based security testing, abuse cases, security requirements, security operations

1. list the knowledge catalogs that form the foundation of software security comprehension

Principle = general security wisdom

Guideline = recommendation at semantic level

Rule = recommendation at syntactic level

Vulnerability = result of software defect

Exploit = particular instance of an attack

Attack pattern helps identify/qualify the risk that a given exploit will occur

Historical risk identified in actual software development effort

1. apply the risk management framework for a specific problem description

Understand business context (goals)

Identify business and technical risks

Synthesize and prioritize the risks

Define risk management strategy mitigation

Carry out required fixed and validate they are correct

1. design a custom software development lifecycle outline that incorporates risk management, touchpoints, and knowledge

Knowledge during software development(follow the principle, guideline, rule), touchpoints during testing(review, analysis, testing), risk management(understand, identify, prioritize risk), knowledge(vulnerability, exploit, attack pattern, historical risk)

1. describe measurements that could be taken to ensure data-driven decisions about risk management

# of risks identified, identified and mitigated, identified and grouped by priority

### Fundamentals of Privacy

1. describe ways in which information is collected

Surveillance = watching, listening to individuals’ behaviors

Interrogation = questioning/probing for information

1. describe ways in which information is processed（AI ISE)

Aggregation = combine pieces to form a large understanding

Identification = link infor to specific individual

Insecurity = carelessness with storing/protecting collected infor from abuse/misuse

Secondary use = use infor for a different purpose

Exclusion = failing to inform individual what infor was collected

1. describe ways in which information is disseminated

Breach of confidentiality = breaking agreement to keep infor confidential

Disclosure = revealing truthful infor about individual in a way that affects others’ judge

Exposure = revealing nudity/grief/bodily functions

Increased accessibility = making infor more readily accessible

Blackmail = threatening to disclose infor

Appropriation = use infor to serve interests of another

Distortion = disseminating false/misleading infor

1. classify an action as an invasion of privacy or not

Does not need to involve personal information

Intrusion = disturb individual’s tranquility/solitude

Decisional interference = government’s invasion into an individual’s decisions regarding the private affairs

1. explain the purpose of a privacy policy

How infor is collected, used, disclosed

1. describe the Code of Fair Information practices

Practices for strengthening privacy of consumer infor:

* 1. No secret personal data record system (transparency)
  2. Find out what information is collected and how it’s used (no secondary use)
  3. Able to prevent infor from being used/disclosed without consent (no exclusion)
  4. Correct the record of identifiable information
  5. Create, use, maintain the data as intended, prevent from misuse(integrity)

1. create a privacy policy for a given scenario

Notice: easy to understand, what data will be collected

Consent: choose how the data to be used

Integrity: how company measure data and protect it from misuse

Access: to collected data

1. list examples of protected health information

Def = identifiable health infor that can identify a patient

Eg. demographics, medical infor, financial infor

1. explain the purpose of the Health Insurance Portability & Accountability Act

Protect privacy/security of PHI(protected health infor)

Standardization of electronic data interchange

1. describe the technical controls mentioned by the Health Insurance Portability & Accountability Act

Access control(only authorized persons access e-PHI)

Audit control(must record/examine access in information system)

Integrity control(e-PHI is not improperly altered/destroyed)

Transmission security(guard against unauthorized access to e-PHI)

1. explain the purpose of the Gramm-Leach Bliley Act

Protect consumers’ personal financial infor held by financial institutions

1. describe the Financial Privacy Rule of the Gramm-Leach Bliley Act

Governs collection/disclosure of customers’ nonpublic personal financial infor

Requires companies to give privacy notices that explain institutions’ infor sharing practices

Consumers have the right to limit some(not all) sharing of their infor

1. explain how the Pretexting Provisions of the Gramm-Leach Bliley Act relate to software security

Pretexting provision = prohibit pretexting(access private infor with false pretenses)

=> accountability

1. describe techniques for protecting user identities

Usernames/ids should be case sensitive

Usernames should be unique and private

1. describe techniques for protecting identities of data subjects

MFA, hash password, CAPTCHA, session timeout

1. explain how reidentification can affect the security and privacy of data in software

Re-identification = matching anonymous data with publicly available information to discover the individual to which the data belongs to

Institutions may release the data after de-identification process.

Taxonomy of privacy: collection, processing, dissemination, invasion of privacy

Privacy breach = activity that creates harm by disrupting activities

### Vulnerability Rankings

1. list the most common web application security vulnerabilities

Injection, broken authentication, XSS, broken access control, security misconfiguration, sensitive data exposure, insufficient attack protection, CSRF, using component with known vulnerabilities, underprotected APIs

1. summarize the evolution of the list of the most common web application security vulnerabilities

Some risks remains <= they are hardest to prevent; complicated to protect

Some risks are drooped <= find the tools or designed to avoid them;

Some risks are added <= new ideas explored; company reputation(sensitive data)

1. use the OWASP website to find information about the most common web application security vulnerabilities

Injection, broken authentication, XSS, security misconfiguration, sensitive data exposure, insufficient attack protection, CSRF, using component with known vulnerabilities, underprotected APIs

1. describe the most common software weaknesses

SQL Injection, OS command injection, XSS

1. use the CWE/SANS website to find information about the most common security weaknesses

Insecure interaction between components:

SQL/OS injection, XSS, CSRF, open redirect, Unrestricted Upload of File with Dangerous Type

Risky resource management

Classic buffer overflow, path traversal, download code without integrity checks, Use of Potentially Dangerous Function, Incorrect Calculation of Buffer Size, Integer Overflow

Porous defenses

missing/incorrect authentication, Missing Encryption of Sensitive Data...

1. compare and contrast the CWE/SANS Top 25 with the OWASP Top 10

Similarities like injection, missed authentication, sensitive data, CSRF

### Injection

1. explain how software may be vulnerable to injection attacks

Execution of system command to access/modify/delete data

1. create user input that produces an injection attack

Password = ‘ OR ‘1’ = ‘1

1. describe techniques for mitigating injection attacks in software

Never trust user input

Static analysis tool

Implement security checks on client side and server side

blacklist/whitelist user input

Sanitize user input(convert to string instead of command)

Prepared statement(setter method)

Database framework(Hibernate)

### Cross-Site Scripting

1. explain how software may be vulnerable to cross-site scripting attacks

Control appearance of a site, transfer sensitive data

1. create user input that produces a cross-site scripting attack

Reflected XSS = <script> alert(“attacked”)</script>

1. describe techniques for mitigating cross-site scripting vulnerabilities in software

Never trust user input

Static analysis tool

Implement security checks on client side and server side

blacklist/whitelist user input

Sanitize user input(convert to string instead of command)

Use encoding libraries(make sure rules for sanitizing user input)

### Cross-Site Request Forgery

1. explain how software may be vulnerable to cross-site request forgery attacks

Trick a browser into performing undesired request to website on behalf of logged-in users

When a web app doesn’t sufficiently if the request is valid

1. create a cross-site request forgery attack for a web application

<img src="http://myserver.com/updateProfile?email=john@abc.com" width="1" height="1" />

1. describe techniques for mitigating cross-site request forgery vulnerabilities in software

Use anti-CSRF tokens

Check headers(HTTP referrer and HTTP origin)

Require multiple steps for dangerous/critical operations

### Other Input Validation Vulnerabilities

1. describe how unrestricted file uploads can affect the security of software

Upload file that can be automatically processed

1. describe how untrusted inputs in security decisions can affect the security of software

Use security mechanism that relies on the input value, and the value can be modified maliciously in a way that bypasses the security mechanism

1. describe how path traversal can affect the security of software

External input to build a pathname that identifies a restricted file -> resolve the location

1. describe how uncontrolled format strings can affect the security of software

Buffer overflow or access illegal memory location

1. describe how buffer overflow attacks can affect the security of software

Execute arbitrary code outside the authorized scope

1. describe techniques to mitigate buffer overflow vulnerabilities

Use a language with protection

Use compiler extensions to help identify and capture possible issues

Validate user input

Run software with least privileges

1. describe how integer overflow attacks can affect the security of software

When the calculated integer is used for execution control, security checks

1. describe techniques to mitigate integer overflow vulnerabilities

Choose data type carefully

Code inspections to identify potential integer overflow

Check for out-of-bounds values

### Improper Authentication

1. describe how broken or missing authentication affects the security of software

Attackers can impersonate other users and gain access to unauthorized functionality

1. describe techniques for mitigating attacks resulting from broken or missing authentication

Don’t pass SessionID in URL

Session timeout

Hash password

Encrypt essential data

Use CAPTCHA to prevent automated attack

Multi-factor authentication

1. describe how allowing excessive authentication attempts affects the security of software

Repeatedly try to guess a user’s login credentials

1. describe techniques for mitigating attacks resulting from improper restriction of excessive authentication attempts

Lockout account after a small number of failed authentication attempts

lock/wait period to slow-down the repeated authentication attempts

CAPTCHA

### Improper Access Control

1. describe how broken or incorrect authorization affects the security of software

Obtain unauthorized access to delete account

1. describe techniques for mitigating risks associated with broken or improper authorization

Access control checks are performed related to business logic

No display links and buttons to unauthorized functions

Least privileges are given

1. describe how execution with unnecessary privileges affects the security of software

Amplify the consequences

1. describe techniques for mitigate risks associated with unnecessary privileges

Lowest privileges required

Privileges are dropped when a user don’t need it

Harden OS with latest patches

### Improper Data Exposure

1. describe how sensitive data exposure affects the security of software

Unauthorized access to sensitive data and resources

1. describe techniques for mitigating risks associated with sensitive data exposure

Encrypt to secure data at rest and in transit

Don’t store sensitive data that is not necessary for the purpose of the software

1. describe how improper encryption affects the security of software

Use a risky cryptographic algorithm to secure sensitive data

1. explain the basics of how encryption and hashing secure data

Encryption = encoding data so its meaning is not obvious (to protect data in transit)

Hashing = mapping between arbitrary length input and a fixed length output (to verify data hasn’t been altered)

1. describe techniques for mitigating risks associated with improper encryption

Define sensitive data and how to protect them

Industry approved cryptography method and techniques

Up-to-date cryptographic algorithms to encrypt data

## Exam 1 answer

### Multiple Choice

1. Gary McGraw considers ( ) as the most important security touchpoint developers should incorporate into the software development lifecycle
   1. risk analysis
   2. security requirement
   3. penetration testing
   4. code review with a tool
2. Injection attacks can occur when ( ) is/are embedded in an instruction stream
   1. database query commands
   2. programming language commands
   3. operating system commands
   4. all of the above
3. Use of ( ) is one way to mitigate against ( ) vulnerabilities
   1. prepared statement; XSS
   2. prepared statement; injection
   3. HMACs; injection
   4. direct object references; CSRF
4. if a vulnerability allows data to be modified while in transit across a network, then that vulnerability weakens which security objective?
   1. confidentiality
   2. integrity
   3. accountability
   4. privacy
5. during a lab activity with DVWA, we first performed a ( ) attack to obtain a user’s session ID so that we could then execute a ( ) attack
   1. XSS; CSRF
   2. CSRF; XSS
   3. SQL injection; XSS
   4. SQL injection; CSRF
6. The first step of risk management framework is to ( )
   1. define the risk mitigation strategy
   2. identify the business and technical risks
   3. prioritize and rank the risks
   4. understand the business context
7. Information can be collected through ( )
   1. aggregation and blackmail
   2. surveillance and coercion
   3. interrogation and surveillance
   4. interrogation and exploitation
8. Privacy policies should, at a minimum, address ( )
   1. anonymity/privacy, choice/consent, integrity/security, possession/protection
   2. consent/disclosure, collection/processing, dissemination/invasion, integrity/security
   3. integrity/security, accountability/privacy, identification/authorization
   4. notice/awareness, choice/consent, integrity/security, access/participation
9. Which of the following vulnerabilities appeared in the OWASP Top 10 web application security vulnerabilities in 2004 and remains on the list in 2017?
   1. improper use of encryption
   2. denial of service
   3. injection
   4. inadequate logging
10. All of the following are categories of vulnerabilities in the CWE/SANS Top 10 except:
    1. porous defense
    2. perimeter defense shortfalls
    3. insecure interaction between component
    4. risky resource management

### Matching

(Confidentiality) Degree to which data is disclosed only as intended

(Accountability) Degree to which actions affecting software assets can be traced to the actor responsible for the action

(security vulnerability/design flaw) A mistake or weakness in the way software is designed

(risk management) Identifying, ranking, tracking, understanding, and mitigating threats throughout the software development lifecycle

(exploit) A particular instance of an attack on a computer system

(privacy) the claim of individuals, groups and institutions to determine for themselves, when, how and to what extent information about them is communicated to others

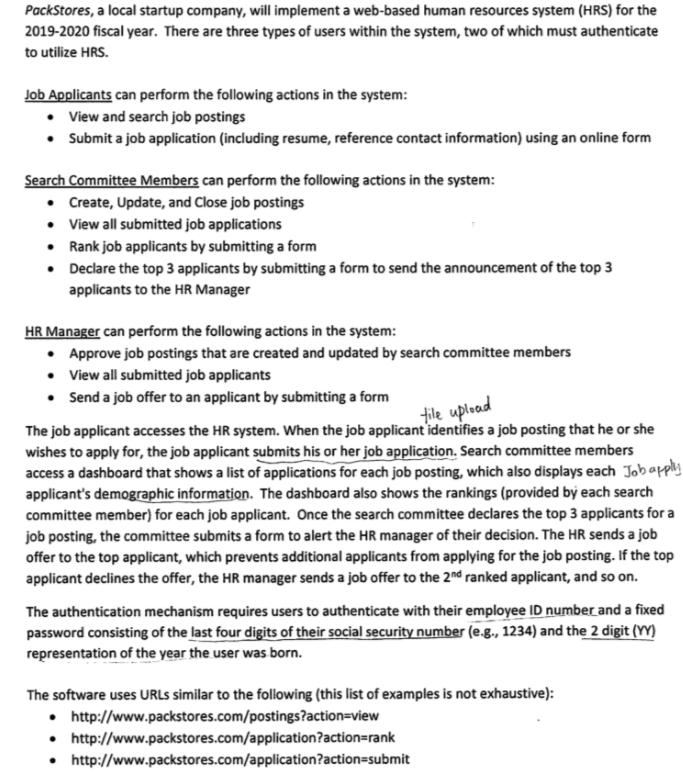
(disclosure) revealing truthful information about an individual in such a way that affects the way others judge the individual

(distortion) disseminating false/misleading information about an individual

an error, flaw, failure, or fault in a piece of software aggregation/exposure

(touchpoints) a set of best practices that software teams should utilize at specific phases of software development

### Short Answer

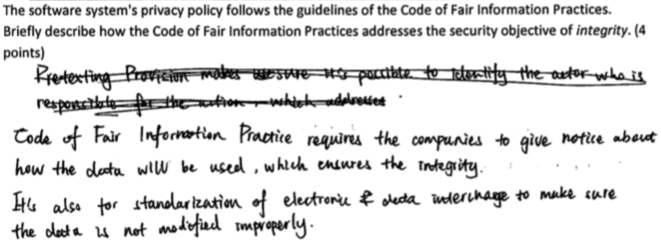


List 4 specific locations in the software system software where input should be validated

1. Job applicants submit the job application with online form
2. Any type of users tries to authenticate or login to the system
3. Job applicants search job postings
4. Committee members submit the form with ranked applicants

List 5 different techniques the software development team could use to mitigate input validation vulnerabilities

1. statistic tools
2. security checks on server and client sides
3. sanitize user input
4. create blacklist and whitelist of user input

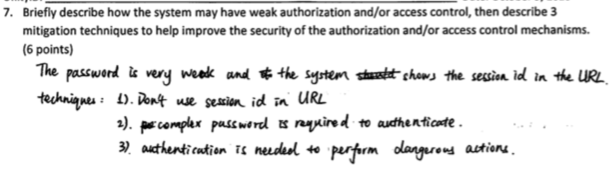


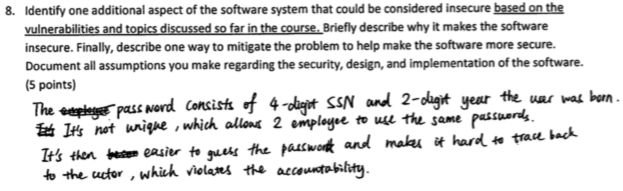
Briefly describe 3 ways the authentication system should be improved based on the OWASP guidelines for passwords and authentication mechanisms

1. Multi-factor authentication should be used
2. Lockout after a small number of failed attempts
3. Complex password is required

Briefly describe how you could perform a cross-site scripting attack in the web application. Give a specific example using the information contained in the software description. Stored/reflected?

<script>alert(“you’re attacked”)</script>. It’s a stored attack



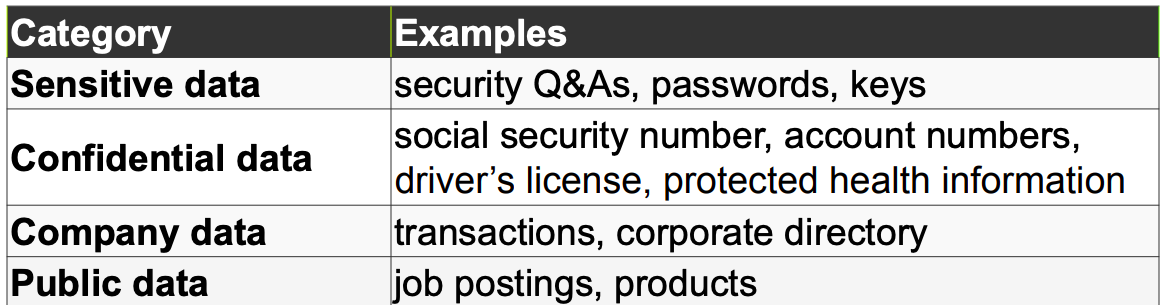


## Exam 2

### Improper Data Exposure

1. describe how sensitive data exposure affects the security of software

Unauthenticated access to sensitive data and resources =>



1. describe techniques for mitigating risks associated with sensitive data exposure

Industry-standard encryption to secure data at rest and in transit

Not store sensitive data if it’s not necessary for the system

1. describe how improper encryption affects the security of software

Fail to secure sensitive data at rest or in transit

Broken cryptographic algorithm

Not hash with a salt(help reduce the risk of cracking a hash, add a salt (a random value) to the value to be hashed)

1. explain the basics of how encryption and hashing secure data

Encryption: encoding message so the meaning is not so obvious

Decryption: transforming encrypted message to normal text

Hashing: mapping between an arbitrary length input and a fixed length output. It’s one-way function

1. describe techniques for mitigating risks associated with improper encryption

Identify sensitive data and how to protect them

System should be designed in a way allowing the cryptographic algorithm to be updated

Industry approved cryptography methods

Ensure encryption is properly integrated into the system

Up-to-date cryptographic algorithm

### Use of Dangerous Functions

1. describe how use of third-party components can affect the security of software

Use potentially dangerous functions incorrectly may unintentionally introduce vulnerabilities into the software

1. describe techniques for mitigating risks associated with the use of third-party components

* Latest version of third-party libraries
* Identify all components and versions including all dependencies

1. describe common software misconfigurations that affect the security of software

Missing patches, misconfigured/disabled security features, default accounts, unnecessary/unused service/features,

use of hard-coded credentials(pw, cryptographic keys) helps attackers bypass authentication mechanisms.

* Encrypt and store credentials in protected configuration files
* Perform authorization checks before performing sensitive functionality
* Ensure access control mechanisms limit access to features that use credentials

Insufficient Attack Protection: serves a nearly constant stream of requests from legitimate and malicious users

* Detect both manual and automated attack
* Response to attack(logging and notification)
* Patch the system as quickly as possible

Underprotected APIs: security vulnerabilities can exist in APIs

* Follows the mitigating method for the security vulnerabilities

1. Mitigations see 3

## Exam 2 answer

### Multiple Choice

1. Hashing is ( )
   1. a type of encryption algorithm
   2. an algorithm that can be reversed
   3. a type of decryption algorithm
   4. a one-way function
2. All of the following are ways to mitigate risks associated with improper authentication except….
   1. pass session IDs in the URL
   2. hash passwords with a salt
   3. encrypt security questions
   4. use CAPTCHAs
3. ( ) access control permits the granting and revoking of privileges to be responsibility of the people who “own” the data being accessed
   1. role-based
   2. mandatory
   3. discretionary
   4. many-to-one
4. Which of the following is not a security practice for software development?
   1. perform static analysis
   2. establish design requirement
   3. perform dynamic analysis
   4. perform initial security review at release
5. Use of OWASP’s input validation frameworks best supports which design principle?
   1. secure by default
   2. separation of privilege
   3. don’t reinvent the wheel
   4. secure the weakest link
6. At what point should a software development team create abuse/misuse cases in the software development lifecycle?
   1. as soon as they have requirements
   2. after implementing the software
   3. during system testing
   4. after a security/privacy breach happens
7. ( ) combined with ( ) typically indicates high security risk
   1. Good code quality; low attack surface
   2. Bad code quality; low attack surface
   3. Bad code quality; high attack surface
   4. none of the above
8. All of the following are techniques to reduce attack surface except…
   1. add additional administrative users
   2. disable/close unused ports
   3. disable/close unused processes
   4. reduce the number of dynamic web pages
9. Abuse/misuse cases should be used for…
   1. validating mitigation strategies
   2. documenting how the software reacts to illegitimate use
   3. determining impacts of changes in impact of an attack
   4. comparing and ranking security risks
10. Because of their predictable conventions, RESP APIs are particularly susceptible to …
    1. improper authorization
    2. improper authentication
    3. improper data exposure
    4. injection attacks
11. What design principle is demonstrated by the phrase, “Ensure privileges are dropped when a user no longer needs the privilege to carry out job-related tasks”?
    1. separation of privilege
    2. least privilege
    3. secure by default
    4. security through obscurity
12. Data should be encrypted and protected …
    1. in storage
    2. in storage and in transit
    3. in transit
    4. none f the above
13. Use ( ) when you want to compare a value but can’t store the plain representation
    1. public key encryption
    2. hashing
    3. private key encryption
    4. decryption
14. Hashing can be “cracked” through all of the following, except …
    1. brute-force attacks
    2. dictionary attacks
    3. rainbow/lookup tables
    4. salting
15. All of the following are examples of security misconfigurations, except …
    1. enabling logging mechanisms
    2. missing patches
    3. running unnecessary services/features
    4. general, infeasible, and measurable

### Multiple Choice

### Short Answer

## Exam 3

### Secure Software Development Lifecycle

1. list the phases of the security development lifecycle

Training, requirement, design, implementation, verification, release, response

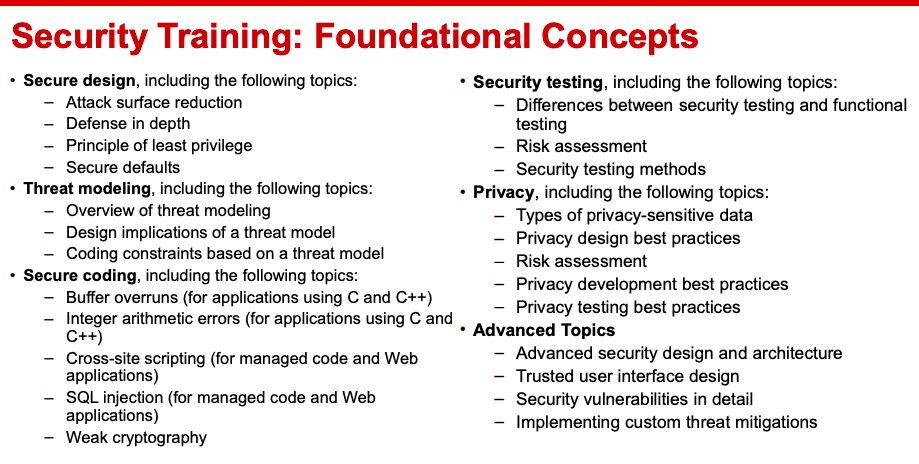
1. describe how and why the security development lifecycle was first developed

A software development security assurance process consisting of security practices

Affects all steps in the lifecycle and the development culture

1. describe security practices and techniques that can be incorporated into each phase of the security development lifecycle

Training: P1(Core Security Training)



Requirement:

* P2(Security and Privacy Requirements),
* P3(Create quality gates/bug bars),
* P4(Security and Privacy Risk Assessment)

Design:

* P5(Establish design requirements EG. consider security principles)
* P6(Attack surface analysis/reduction),
* P7(Threat modeling)

Implementation:

* P8(Use Approved Tools)
* P9(Deprecate Unsafe Functions)
* P10(Perform Static Analysis)

Verification:

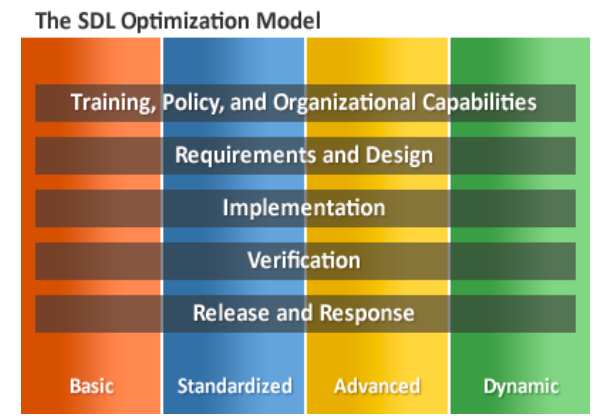
* P11(Perform Dynamic Analysis),
* P12(Fuzz Testing),
* P13(Attack Surface Review)

Release:

* P14(Incident Response Plan),
* P15(Final Security Review),
* P16(Release/Archive)

Response: P17(Execute Incident Response Plan)

1. describe the security development lifecycle optimization model



1. describe qualities of software and organizations at each level of the SDL optimization model

Optimization level 1: Basic

Overall security health of applications and services is unknown:

* Manual, voluntary, or localized processes
* Nonexistent, undefined, or unenforced policies and standards for secure architecture, design, development, testing, compliance, and other common security practices.
* Reactive practices driven by external pressures.

Optimization level 2: Standardized

* Able to assess the security and privacy risk of new projects
* Able to select the best candidates for implementing security and privacy practices into the development lifecycle.
* Security and privacy practices are applied to a few pilot projects and there is only tacit executive support for the efforts.
* Much of the effort is still spent at later phases of the lifecycle and in security response.

Optimization Level 3: Advanced

* All new and high-risk projects fall under the mandate for the SDL practices and quality gates.
* Security and privacy practices are integrated throughout the software development lifecycle
* Security testing guidelines are in place
* Tools are effectively used to reduce costs.
* Security response is rapid and controlled
* Practice of applying security and privacy efforts at earlier lifecycle phases helps to reduce the overall cost of producing secure software

Optimization level 4: Dynamic

* Organizational training goals are in place
* Mandate for secure software at all applicable projects across the entire organization.
* Teams have internalized the development of secure software
* Security efforts can be focused on proactive innovation and tool customization, in addition to reactive incident response, and products are known for security excellence in the marketplace.

### Security Requirements

1. document functional security requirements for a software system

A condition or capability needed in the system to control or limit the fulfillment of requirements

1. explain whether a given functional security requirement is SMART

Specific + Measurable + Actionable(what needs to be done) + Realistic(time, budget) + Timely(prioritization)

1. identify security objectives that are relevant for a given functional requirement

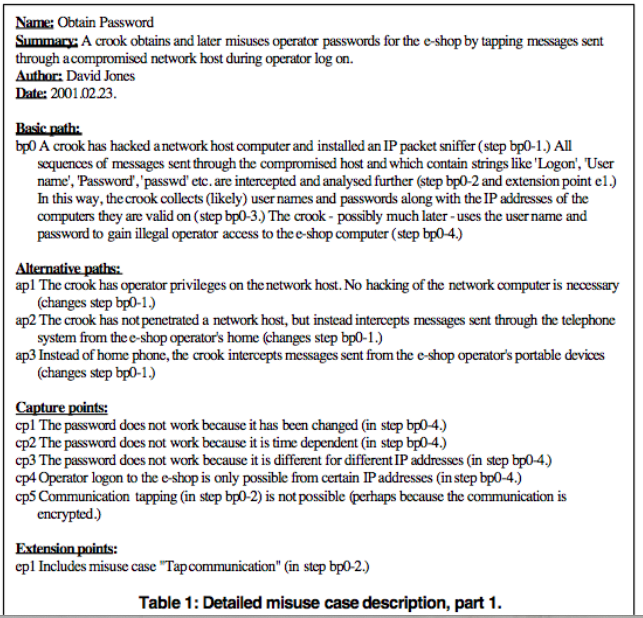
Confidentiality, integrity, accountability, identity, availability, privacy

1. document functional security requirements that are implied by a given functional requirement

Positive, functional behavior related to security

Testable

1. document an abuse or misuse case description



1. draw a use case diagram that includes abuse and misuse cases that threated the software's use cases

(good)<---threatens---(misuse)

1. draw a use case diagram that includes use cases to mitigate existing abuse and misuse cases

(method) --mitigates→(misuse)

1. explain what adverserial thinking is and how it helps security professionals mitigate potential threats

Developers are able to mentally think how things can go wrong or break.

### IEEE Design Principles 1-5

1. explain the following concepts:

* Defense in depth: multiple defensive countermeasures to discourage attackers(people with policies, technology with encryption, operations with key management)
* Separation of privilege: permission is not granted with a single condition(password + role)
* Secure by default: default setting should as secure as possible (no default password)
* Fail securely: don’t disclose data, provide access, or provide too much information when the failure occured.
* Least privilege: the least privilege is given according to the need.

1. explain each of the following concepts:

* discretionary access control: the owner group can grant/revoke access privileges. (Google drive owner determines the access control)
* mandatory access control: security strategy that restricts the ability individual resource owners have to grant or deny access to resource objects in a file system. (highly sensitive government information)
* role-based access control: the access is given based on the roles.

1. create an access control matrix for a piece of software
2. explain which type of access control would be more suitable for a piece of software
3. describe techniques for mitigating denial of service attacks

Appropriate authentication : an entity verifies that another entity is who or what it claims to be

Appropriate authorization : if the authenticated entity has authority to access to the resource being requested

Filtering : inspecting data as it’s received and making a decision to accept or reject the input

Throttling : limiting the number of requests to your system, particularly with anonymous requests; intrusion-detection systems

Quality of Service provide preferential treatment for specific types of traffic

1. explain how trust impacts the security of software

Trust should always be closely held, never loosely given and never by default.

### IEEE Design Principles 6-10

1. describe common pitfalls when applying cryptography in software systems

* Rolling your own cryptographic algorithms or implementations
* Misuse of libraries and algorithms
  + Incorrect assumptions
* Poor key management
  + hard-coding keys into software (often observed in embedded devices and application software)
  + failure to allow for the revocation and/or rotation of keys
  + use of cryptographic keys that are weak (such as keys that are too short or that are predictable)
  + weak key distribution mechanisms
* Randomness that is not random
  + Need random numbers with strong cryptographic randomness properties
  + Cannot reuse random numbers
* Failure to centralize cryptography choice within a team/organization
  + Different cryptographic algorithms often don’t interact nicely.

1. describe techniques to help identify and manage sensitive data

Identify sensitive data: create a policy that identifies diff levels of classification

Manage:

* Procedures must be provided but are changeable over time(policies, regulation, user expectation)
* Data sensitivity can be context-sensitive

Things to consider:

* Access control mechanisms (including file protection mechanisms, memory protection mechanisms, and database protection mechanisms)
* Cryptography to preserve data confidentiality or integrity
* Redundancy
* Data at rest
* Data in transit

1. describe an example that highlights competing concerns with usability and security

2FA

Usability = easy to use(easy password or fewer levels to check privilege/authentication)

1. explain the components of a software system's attack surface

Sum of all the paths for data into and out of the application

All valuable data used in the system

1. describe techniques to reduce a software system's' attack surface

* Isolate components as much as possible
* Configure to only open functionality you will use
* If the component cannot be configured to comply with your security policy, don’t use it
* Look at vulnerability history in CVE database
* Maintain up-to-date components
* Maintain a healthy distrust
* Authenticate dataflow
* Consider data coming in untrusted

1. describe the Payment Card Industry Data Security Standard and the types of security objectives that are addressed in the PCIDSS requirements

### Logging

1. explain the difference between logging and auditing

Logging putting information into a log file

Auditing processing information from a log file

1. list user activities that should be logged for a given software system

Access control event(assign/delete access privilege)

Administration event(create accounts, view log entry)

Audit event(malicious activity deleted)

Authentication event(login, session lockout)

User security event(unauthenticated access attempt)

1. explain what information should be included in a user activity log

(5W+H)

Who: User identification, Object affected

Why: Reason for access

Where: Timestamp

When: Source identification, Destination identification

What: Priority, Success/Failure indication, Breach access indication

How: Event description

1. define non-repudiation

Assurance that someone cannot deny something.

1. document functional security requirements for logging mechanisms

(Logging mechanism should log the security related activities with necessary information)

Example: the system shall log every time <subject> performs CRUD(create/read/update/delete) on <resource>

## 

### Attack/Defense Tree

Explain the value of attack trees when building security into software

Risk analysis. “What-if” analysis based on the “how”s

Probabilities of success for various attacks

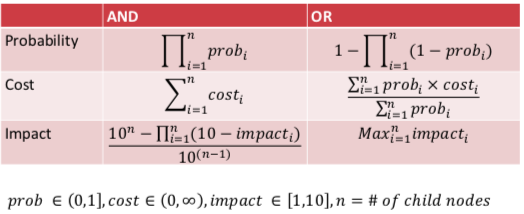
Likelihood of different attacks

Explain the value of defense trees when building security into software

Estimate values for cost, probability, and impact as a component to facilitate risk analysis

1. Logic operations 
2. Impact:

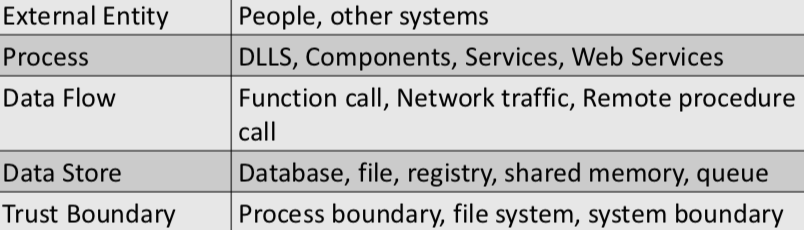
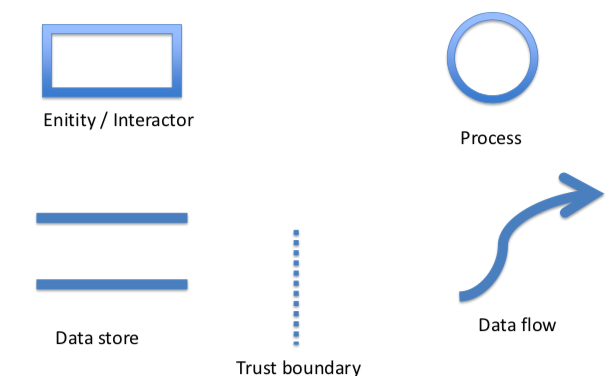
| Numerical Range | Impact Definition |
| --- | --- |
| 1-3 | Minor impact to system. May be a nuisance but is easily detected and/or repaired |
| 4-6 | Moderate impact to system. Confidentiality, integrity, and/or availability of system affected. Requires non-trivial effort to detect and/or repair. |
| 7-9 | Severe impact to system. Significant damage results to system. Considerable effort required to detect and/or repair damage |
| 10 | System completely compromised, inoperable, or destroyed. |

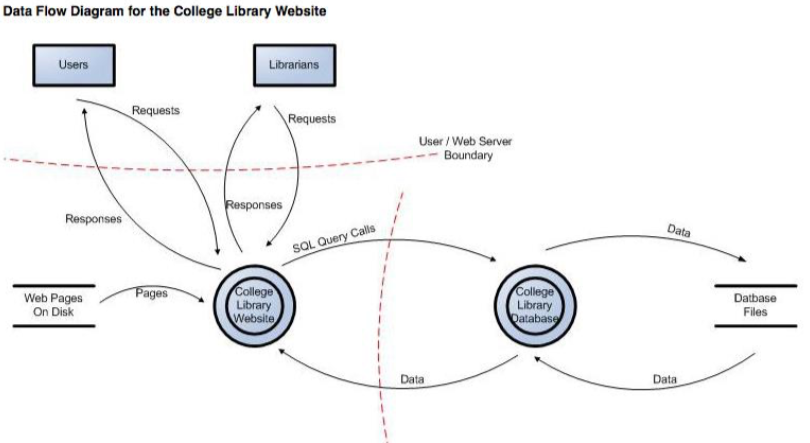
1. Risk = (probability / cost) \* impact; risk(normalized) = log(risk/risk\_min \* 10)
2. Calculation
   1. assign leaf nodes with values
   2. calculate the values

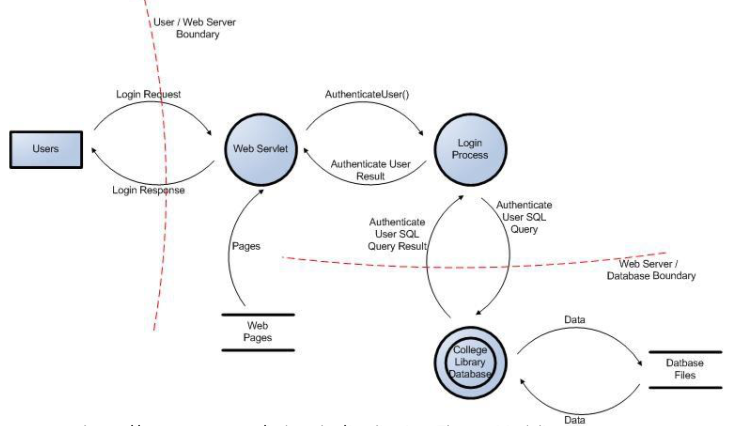


### Threat Modeling

Explain the phases of threat modeling (diagram, identify threats, address threats, validation)



Example 



Identify Threats(STRIDE)

Spoofing(Authentication: impersonating something)

Tampering(Integrity: modifying data/code)

Repudiation(Nonrepudiation: claim to have not performed an action

eg. Nurse/ admin can perform malicious activity and claim to have not performed an action. )

Information disclosure (Confidentiality: exposing information to some unauthorized one)

Denial of Service(Availability: deny/degrade service to users)

Elevation of Privilege(Authorization: gain capacities without proper authorization)



Addressing Threat(META)

Mitigate -> Eliminate(redesign) -> Transfer(another part of the system) -> Accept

| Element | Mitigations |
| --- | --- |
| S | To authenticate principals:  • Basic authentication  • Digest authentication  • Cookie authentication  • Windows authentication (NTLM)  • Kerberos authentication  • PKI systems, such as SSL or TLS and certificates  • IPSec  • Digitally signed packets  To authenticate code or data:  • Digital signatures  • Message authentication codes  • Hashes |
| T | Windows Vista mandatory integrity controls  • ACLs  • Digital signatures  • Message authentication codes |
| R | Strong authentication  • Secure logging and auditing  • Digital signatures  • Secure time stamps  • Trusted third parties |
| I | Encryption  • ACLs |
| D | ACLs  • Filtering  • Quotas  • Authorization  • High-availability designs |
| E | ACLs  • Group or role membership  • Privilege ownership  • Permissions  • Input validation  • Least Privilege |

Prioritize Threats

| Damage | How big would the damage be if the attack succeeded? |
| --- | --- |
| Reproducibility (reliability) | How easy is it to reproduce an attack to work? How reliable is the attack? Does it always work? |
| Exploitability | How much time, effort, and expertise is needed to exploit the threat? |
| Affected Users | If a threat were exploited, what percentage of users would be affected? |
| Discovererability | How easy is it for an attacker to discover this threat? |

Score = sum/5

Validation

Validation of DDF

Able to tell a story without changing the diagram.

Able to tell that story without using words such as “sometimes” or “also”.

We can look at the diagram and see exactly where the software will make a security decision.

Diagram shows all the trust boundaries, such as where different accounts interact.

It covers all UIDs, all application roles, and all network interfaces.

The diagram reflects the current or planned reality of the software.

Able to figure out where all the data goes and who uses it.

Able to see the processes that move data from one datastore to another.

Validation of Threats

We have looked for each of the STRIDE threats.

We have looked at each element of the diagram.

We have looked at each data flow in the diagram.

For each threat, we tried to describe the attack, the context and impact

Validate Information Captured

• Dependencies

– What other code are you using?

– What security functions are in that other code? – Are you sure?

• Assumptions

– Things you note as you build the threat model

Describe who should perform threat modeling within an organization

Building a threat model

Program Manager (PM) owns overall process

Testers

Identify threats in analyze phase

Use threat models to drive test plans

Developers create diagrams

Business analysts

Customers for threat models

Your team

Other features, product teams

Customers, via user education

“External” quality assurance resources, such as pen testers

Identify trust boundaries in a data flow diagram

Trusted/ high code reading from untrusted/low

– Validate everything for specific and defined uses

• High code writing to low

– Make sure your errors don’t give away too much

(What

Consider, document, and discuss security in a structured way (consistency)

• Threat model and document

– The product as a whole

– The security-relevant features

– The attack surfaces

• Assurance that threat modeling has been done well

Why

Produce software that’s secure by design

– Improve designs the same way we’ve improved code

• Because attackers think differently

– Creator blindness/new perspective

• Allow you to predictably and effectively find security problems early in the process)

### Penetration Testing

Phases of penetration testing (information gathering, methodological testing)

Phase 1: information gathering

– Understand environment and the application

– Determine access points, inputs

Goal = discover as much information about the system under evaluation

Phase 2: Testing

– Configuration and deployment management

– Identity, authentication, authorization session management

– Input validation

– Error handling

– Cryptography

– Business logic

– Client side evaluation

Methods: Black-box tests,

Tools: Wireshark, OWASPZap...

Types of penetration testing (targeted recon, social engineering, facilities audit, war driving, dumpster diving)

Targeted Recon.

– Targeted exploitation of vulnerable software.

• Social Engineering

– Hi HelpDesk...I’m Mr. Jones...Can you tell me what my password is?

• Physical facilities audit

find and exploit the vulnerabilities within a company's physical controls and barriers

eg Hmm, I forgot my badge... but there's 200 yards of fence missing on the east side of the center

• Wireless War Driving

Detection of rogue or weakly encrypted AP’s.

• Dumpster Diving

exploration of one's trash that has a main purpose of gathering important information about a person

Eg how much fun can I have in the dumpster...whoops...I’ve found someone’s Tax forms with SSN.

Importance of including penetration testing in software development

Who needs it? banks, financial, government

How to gather information to facilitate penetration testing

– WHOIS

– nmap

– Google Searching and Hacking

– Website browsing

– Social Media

### Usability

Difficulty in balancing security, functionality, and usability

Ex. complex password is secure but hard to remember

Pretexting

Def = attackers focus on creating a good pretext, or a fabricated scenario, that they can use to try stealing their victims’ personal information

Eg. Pretending to be someone authorized to be told it.

Baiting

Def = enticing users for the purpose of tricking them

Eg. Some companies do audits or run fake phishing attacks to assess their employees security practices

Eg. Putting out a basket of free(unknown/untrusted) USB sticks.

Human errors affect the security of software

Slips/lapses at level of skill

– Capture errors: Inattention can cause practiced actions to be performed

EG. People are trained to click “OK” to pop up boxes

EG. Mac makes you type password to install, less automatic

– Post-completion errors:

EG. people leave ATM without card if money comes out first

• Mistakes at level of (following wrong) rules

– OK when URLs that start with impersonated bank name

• www.citibank.secureauthentication.com

• Mistakes at cognitive level

– Just don’t understand the problem

describe why passwords are often considered to be detrimental to usability

Simple password doesn’t secure the account

Complex password is hard to use

"The path of least resistance" principle

（secure by default) The natural and easiest way(do nothing) to do any task should also be the secure way

EG. Default settings of all systems should be secure (“fail-safe defaults”). Users will not read documentation to learn they need to change settings to be secure

Each added inconvenience increases the probability the user will intentionally

decide to use software unsafely

"Appropriate boundaries" principle

（trust boundary）The interface should distinguish objects and actions along boundaries that relate to important issues such as “need to know” or “least privilege”.

– Granularity too detailed and there is a risk roles will overlap or leave out specifications

– Granularity too broad and each role may have more authority than intended

"Explicit authorization" principle

（只有了解权限后才给予user权限，不自动给权限）A user’s authorities must only be provided to other actors as a result of an explicit action that is understood by the user to imply granting

– Follow from (Saltzer’s) principle of least privilege.

– Do not grant privileges by default or implicitly.

– Better to take action to grant new than explicit action to remove unnecessary privilege.

"Visibility" principle

（admin可看capacities）The administrative interface should allow the user to easily review any active authority relationships that would affect security-relevant decisions.

Should be able to see right information for the user to ascertain the limits of what each can do.

Otherwise, a user/application retains an authority undetected and indefinitely once the user has forgotten about the granting privileges.

Particularly important in entitlement-based systems

In an entitlement system, users are given privileges not necessarily based upon roles.

"Revocability" principle

（admin可撤销authorities）The admin interface should allow the user to easily revoke authorities that the user has granted.

– Mistakes giving authority

– Realization that user/application is misguided or malicious

– Revoke does not undo any damages that were done with prior privileges.

"Expected ability" principle

（不能做的不在界面显示）The interface must not generate the impression that it is possible to do something that cannot actually be done.

EG. Interface may imply possible to revoke privileges so authority granted assuming revocation is possible

"Trusted path" principle

（改authority必须是通过trusted path进行）The interface must provide an unspoofable and faithful communication channel between the user and any entity trusted to manipulate authorities on the user’s behalf.

– Example: In Windows, Ctrl-Alt-Del is a non-maskable interrupt that can only be intercepted by the operating system guaranteeing the login window cannot be spoofed

"Identifiability" principle

（不同的object/action代表不同=>identifiable）The interface should enforce that distinct objects and distinct actions have unspoofably identifiable and distinguishable representations.

Continuity = things which are the same should appear the same

Discriminability = things which are different should appear different

"Expressiveness" principle

The interface should provide enough expressive power

(1) to describe a safe security policy without undue difficulty;

(2) to allow users to express security policies in terms that fit their goals.

Configuration settings

Normal course of doing a task

"Clarity" principle

（与安全有关的行为必须先告诉user）The effect of any security-relevant action must be clearly apparent to the user before the action is taken.

– Interface should not be misleading, ambiguous, or incomplete.

– Consequence of security-relevant decisions should be clear so a good decision can be made before an action is taken

### BSIMM

explain how the BSIMM is used in industry to measure an organization's approach to software security

BSIMM as a measuring stick to determine where your approach to software security currently stands relative to other firms.

It tells you what everyone else is actually doing

Practices associated with the following domains of the BSIMM

Governance

Strategy & Metrics - identify gate location + gather necessary artifacts

Compliance & Policy - identify PII obligations

Training - provide awareness training

Intelligence

Attack Models - create a data classification scheme and inventory

Security Features & Design - build and publish security features

Standards & Rqmts - create security standards

SSDL touchpoints

Architecture Analysis - perform security feature review

Code Review - use automated tools along with manual review

Security Testing - drive tests with security requirement and security features

Deployment

Penetration Testing - use external penetration testing to find problem

Software Environment - ensure host and network security basics are in place

Configuration Management - identify software bugs found in operations monitoring and feed them back to development

|  | contact service representative | request turn-by-turn directions | initiate car receiving directions | request stolen vehicle assistance service | make and receive calls with Hands-Free Calling |
| --- | --- | --- | --- | --- | --- |
| driver | yes | yes |  |  | yes |
| passenger | yes | yes |  |  |  |
| service representative |  |  | yes |  |  |
| subscriber |  |  |  | yes |  |