Protecting Information Assets
- Unit #13 -

Computer Application Security

Agenda

- Introduction
- Software development life cycle (SDLC)
- SDLC and security
- Test taking tip
- Quiz

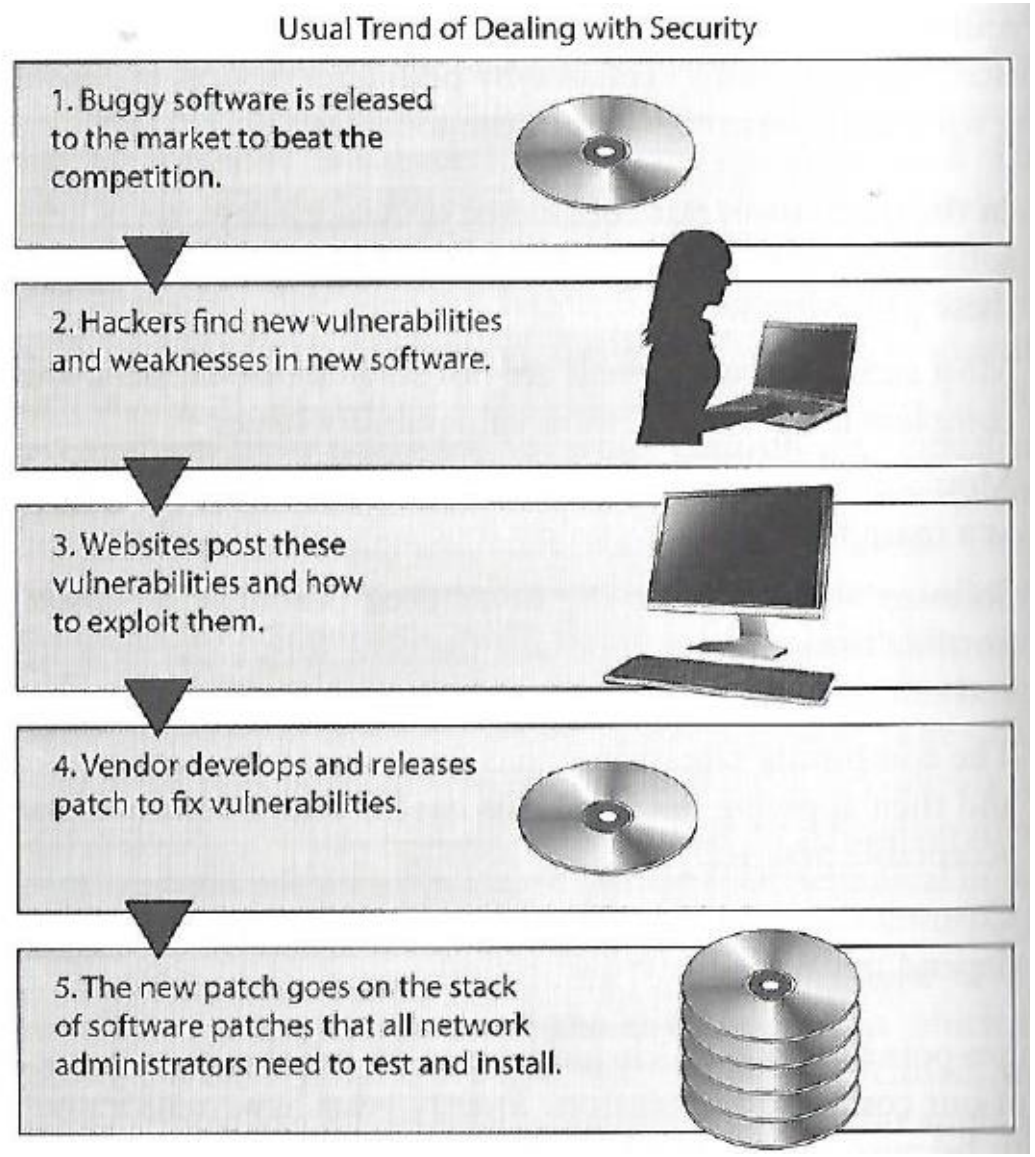
Application Security

As applications become more accessible though the web, cloud and mobile devices,

organizations are being forced to abandon their reactive approach to security and, instead,

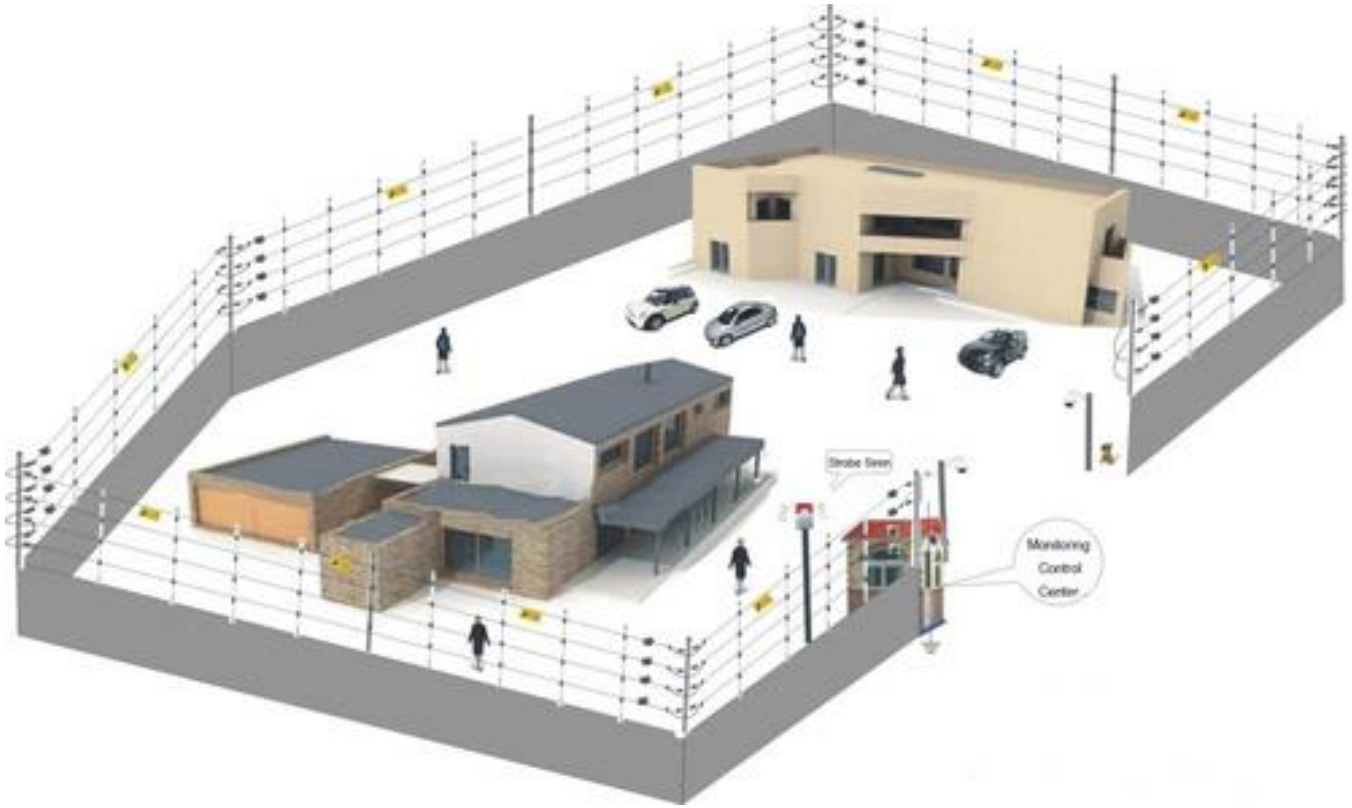
to take a proactive approach by minimizing risk directly in the software they buy, create and use to serve themselves and their customers

Usual trend

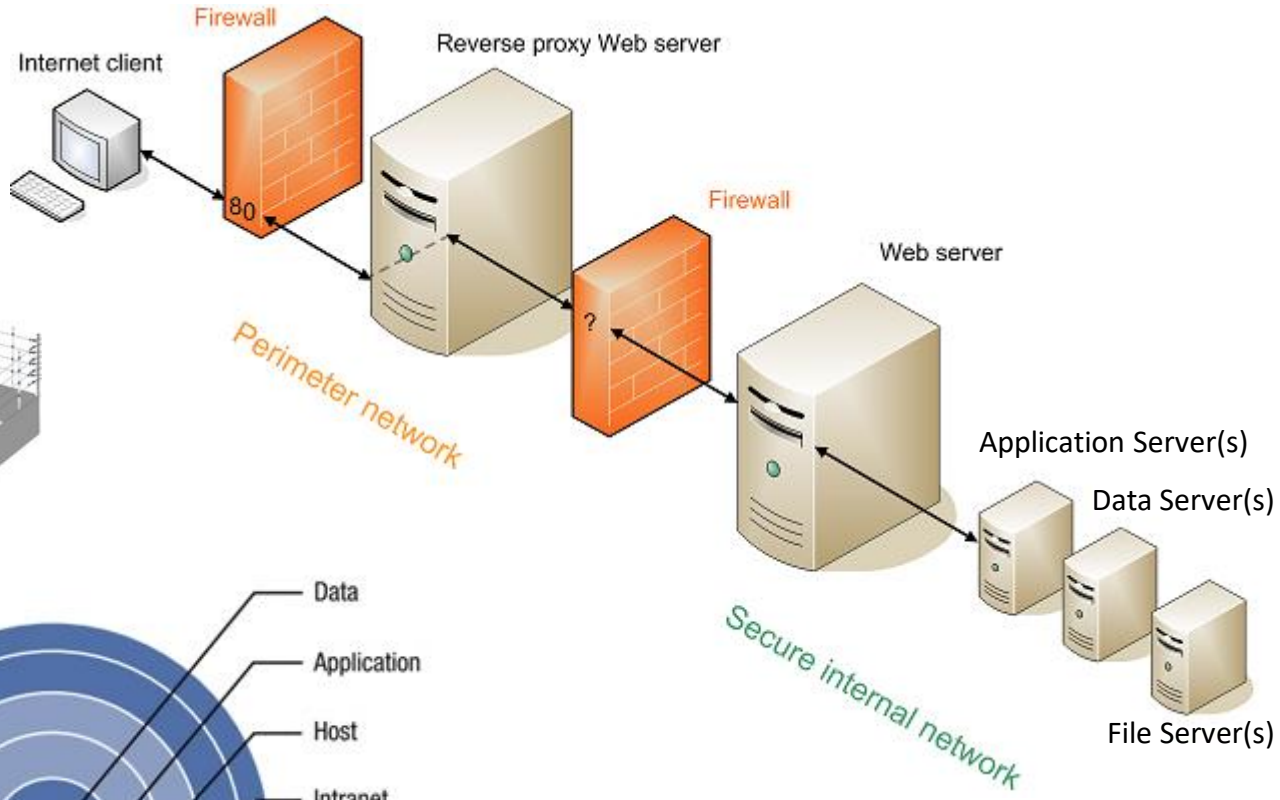


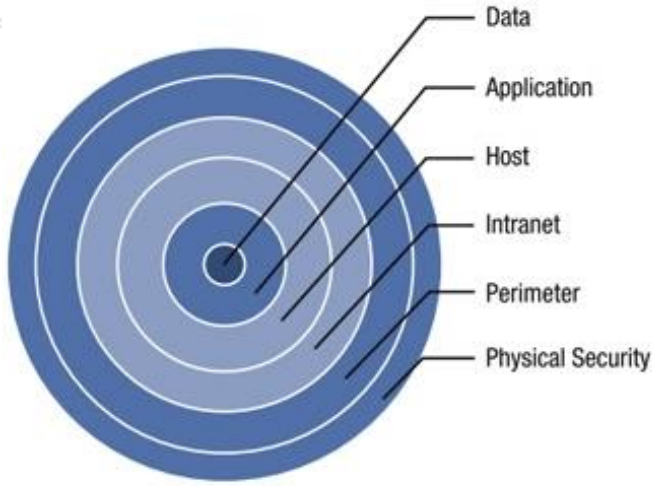
Harris, S. and Maymi F. (2016) CISSP All-in-One Exam Guide, McGraw-Hill Education, p. 1080

Perimeter security solutions are often relied on as a solution to insecure application development practices



Perimeter security solutions are often relied on as a solution to insecure application development practices



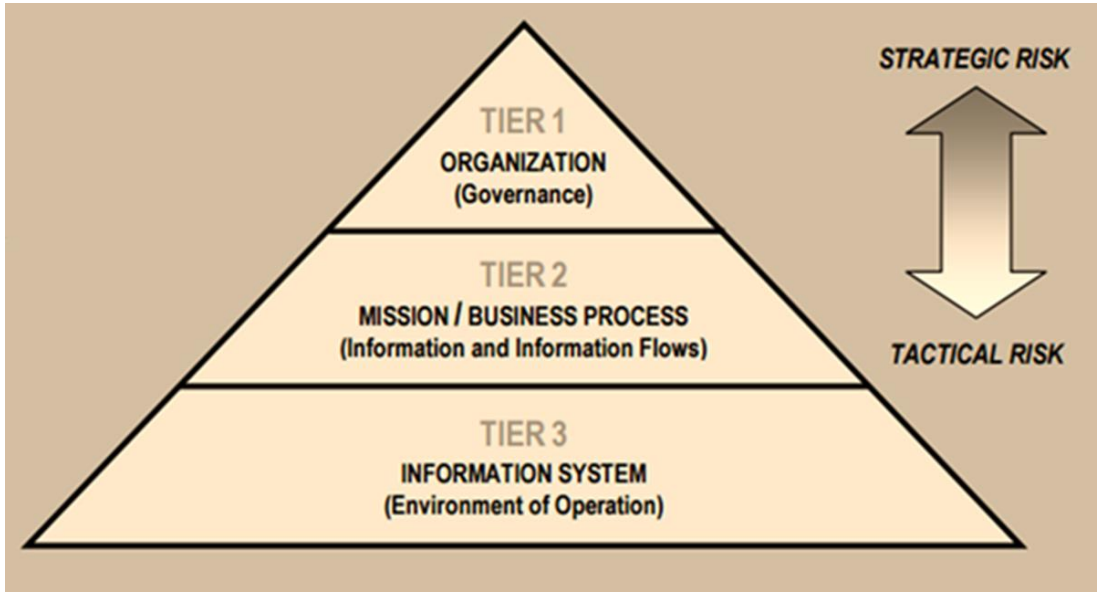


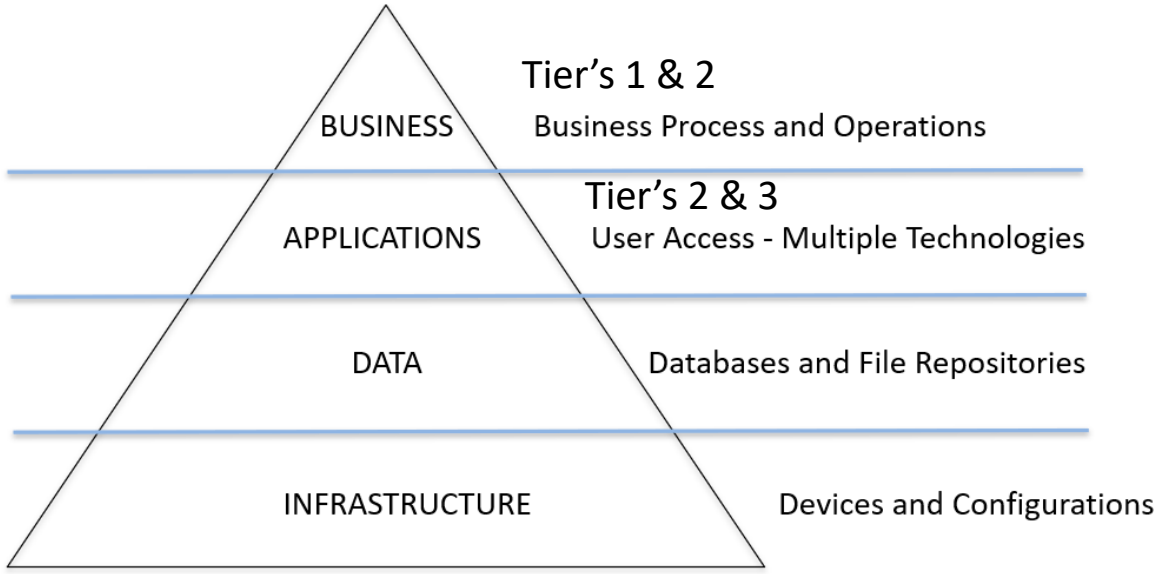
Past and current situation….

- Application developers are not security professionals
  - *Software vendors skip proper security architecture, design and testing steps as they race to beat competitors to market with new features*
- Secure application development practices have not historically been taught in computer science and other academic departments, and are only recently being considered and adopted by developers
- Development projects' scope and budgets focus on functionality, not security
- Security professionals typically not software developers
  - Often lack insight for understanding of software vulnerabilities
- **IT customers…**
  - "Trained" to expect to receive flawed software needing upgrades and patches
  - ***Unable to control flaws in software they purchase, so they rely on perimeter protection***

Security Architecture

Security strategy needs to be a consideration at each level of the architecture





NIST SP 800-39 Managing Information Security Risk
Organization Mission and Information View

Best Practice: Build Security In

Security Architecture	Creation, use and enforcement of System Architecture standards provides the basic building blocks for developing, implementing and maintaining secure applications
Software Development Life Cycle	Attention to security throughout the Software Development Life Cycle (SDLC) is the key to creating secure, manageable applications regardless of platform or technologies
Procurement Standards	Describing the process and detailed criteria that will be used to assess the security level of third party software enables companies to make strategic, security-sensitive decisions about purchased software purchases

Software Development Life Cycle

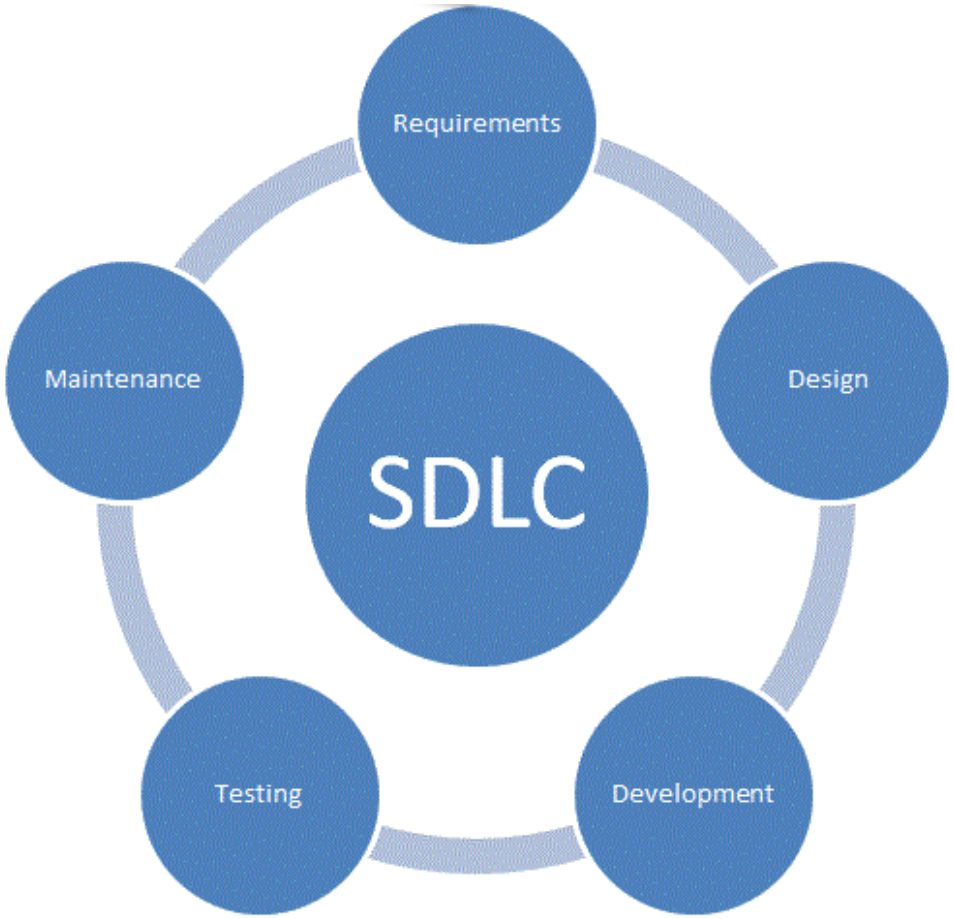
Requirements
- Why the software was created (i.e. goals)
- Who the software was created for
- What the software is intended to do

Design
- Specifications identifying how software and data will be formed to accomplish goals and used to meet requirements

Development
- Programming software code implemented and integrated to meet specifications

Testing-Validation
- Assuring software and data works as planned to meet the goals

Release-Maintenance
- Deploying software and data, and assuring they are properly configured, patched and monitored



Software Development Life Cycle (SDLC)

1. **Requirements analysis**
2. **Design**
3. **Develop (“*make*”) / Implement (“*buy*”)**
4. **Testing/Validation**
5. **Release/Maintenance**

Software Development Life Cycle (SDLC)

1. **Requirements analysis**
   - *Informational, functional, behavioral, and performance specifications…*
2. **Design**
   - *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*
3. **Develop (“*make*”) / Implement (“*buy*”)**
   - *Source code control system, code reviews, daily builds, automated CASE tools…*
4. **Testing/Validation**
   - *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*
5. **Release/Maintenance**
   - *Release testing*

Software requirements specifications documents help support:

Validation

- "Did they build the right application?"
  - In large complex applications it is easy to lose sight of the main goal
  - Does the application/system provide the solution for the intended problem?
  - Are security control specifications included?

Verification

- "Did they build the application right?"
  - Applications can be built that do not match the original specifications
  - Verification determines if the application accurately represents and meets the specifications
  - Verification ensures that security control specifications were properly met

SDLC and Security

1. **Requirements analysis**
   - *Informational, functional, behavioral, and performance specifications…*

   + CIA risk assessment, + Risk-level acceptance,…
2. **Design**
   - *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

   + Threat modeling, + Attack surface analysis,…
3. **Develop (“*make*”) / Implement (“*buy*”)**
   - *Source code control system, code reviews, daily builds, automated CASE tools…*

   + Developer security training, + Static analysis, + Secure code repositories,…
4. **Testing/Validation**
   - *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*

   + Dynamic analysis, + Fuzzing,…
5. **Release/Maintenance**
   - *Release testing*

   + Separation of duties, +Change management,…

Software requirements often specified with…

1. **Information model** – Type and content of information that will be processed and how it will be processed

2. **Functional model** – Tasks and functions the application needs to carry out

3. **Behavioral model** – States the application will be in and transition among

SDLC and Security

Requirements analysis

– *Informational, functional, behavioral, and performance specifications…*

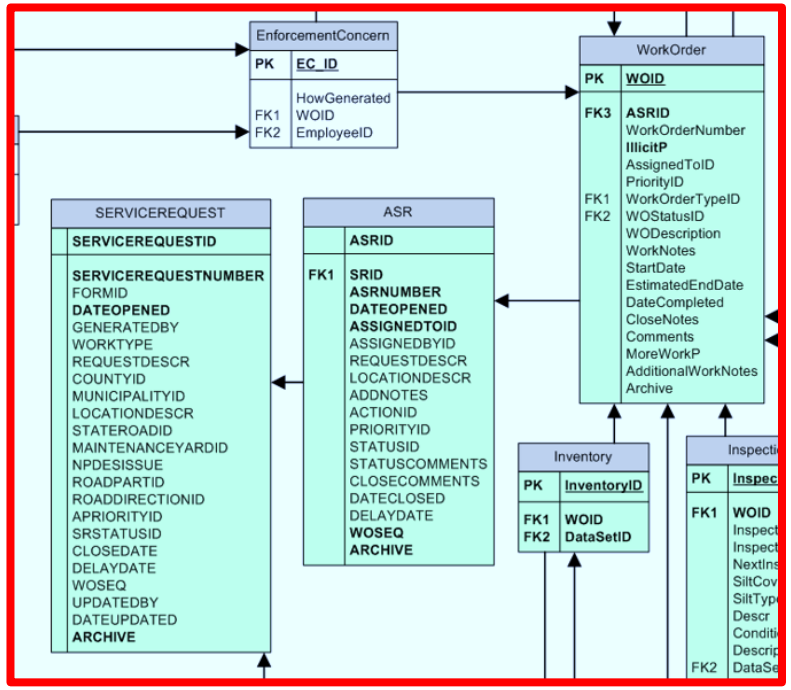
+ CIA risk assessment, + Risk-level acceptance,…

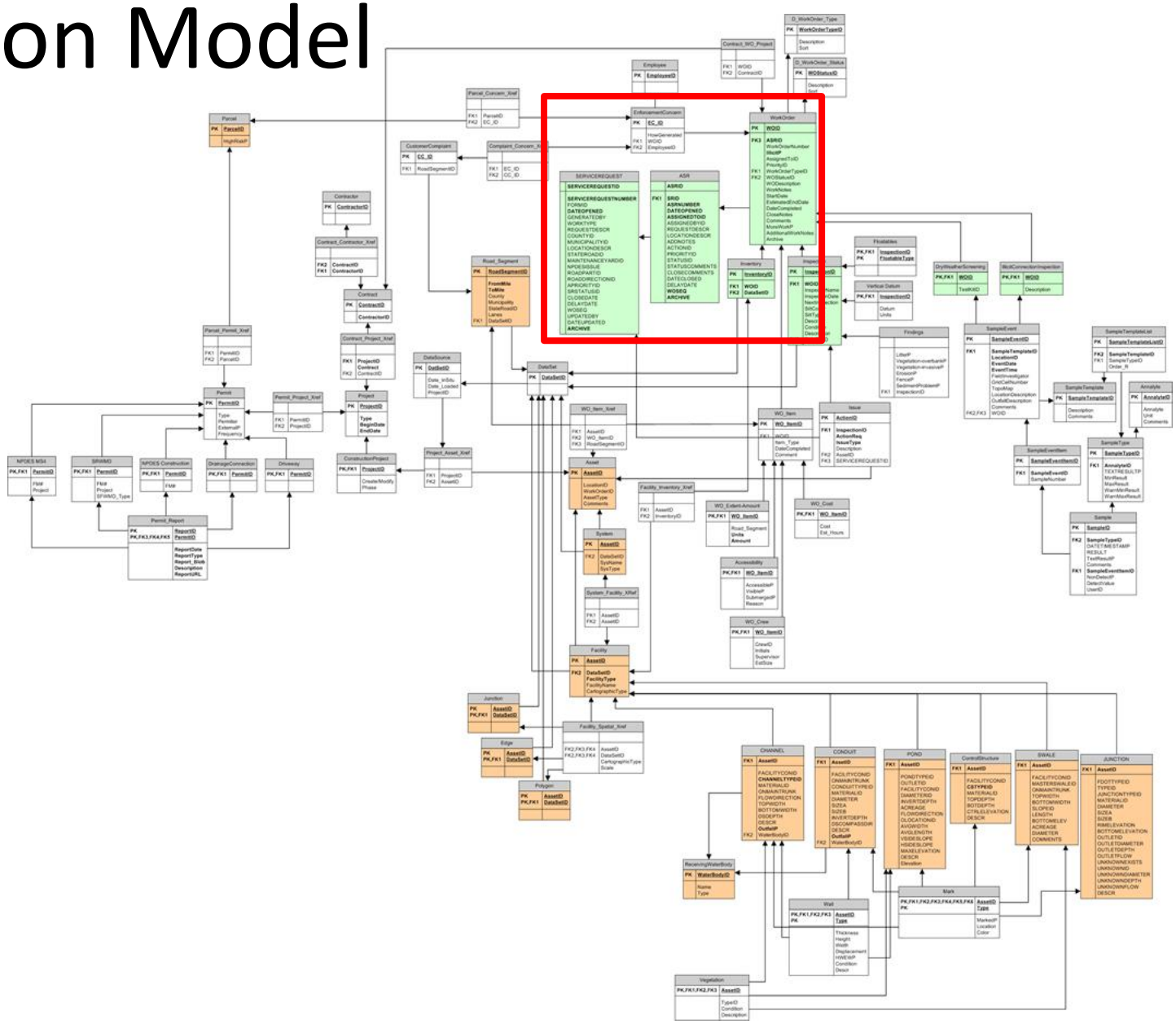
Organisation & relevant process		Information Asset Details									
Operating Unit / Function	**Process name**	**Name of Asset**	**Personal Identifiying Information (PII)** (Y/N)	**Personal Health Information (PHI)** (Y/N)	**Critical Infrastructure Information (CII)** (Y/N)	**Customer Data** (Y/N)	**Organization Data** (Y/N)	**Confidentiality**	**Integrity**	**Availability**	**Categorization**
Thermal Distribution Syster	ChilledWater	TDS	N	N	Y	N	Y	Low	Medium	Medium	
Thermal Distribution Syster	HeatedWater	TDS	N	N	Y	N	Y	Low	Medium	Medium	
Thermal Distribution System		**TDS**	N	N	Y	N	Y	Low	Medium	Medium	**Medium**
Communication	Data	COM	N	N	Y	N	Y	Medium	Medium	Medium	
Communication	Voice	COM	N	N	Y	N	Y	Medium	Medium	Medium	
Communication	Security	COM	N	N	Y	N	Y	High	High	High	
Communication		**COM**	N	N	Y	N	Y	High	High	High	**High**
Public Works	Sewer	Utilities	N	N	Y	N	Y	Low	Medium	Low	
Public Works	Stormwater	Utilities	N	N	Y	N	Y	Low	Medium	Low	
Public Works	Water	Utilities	N	N	Y	N	Y	Low	Medium	Low	
Public Works		**Utilities**	**N**	**N**	**Y**	**N**	**Y**				**Medium**
External	**Parcels**	Parcels	Y	N	N	Y	N	Low	Low	Low	Low

Verification – Information Model

Did they build the application right?

Does it match the data model?





Functional model

Functional Requirements

for

Sewer Outage Notification Application

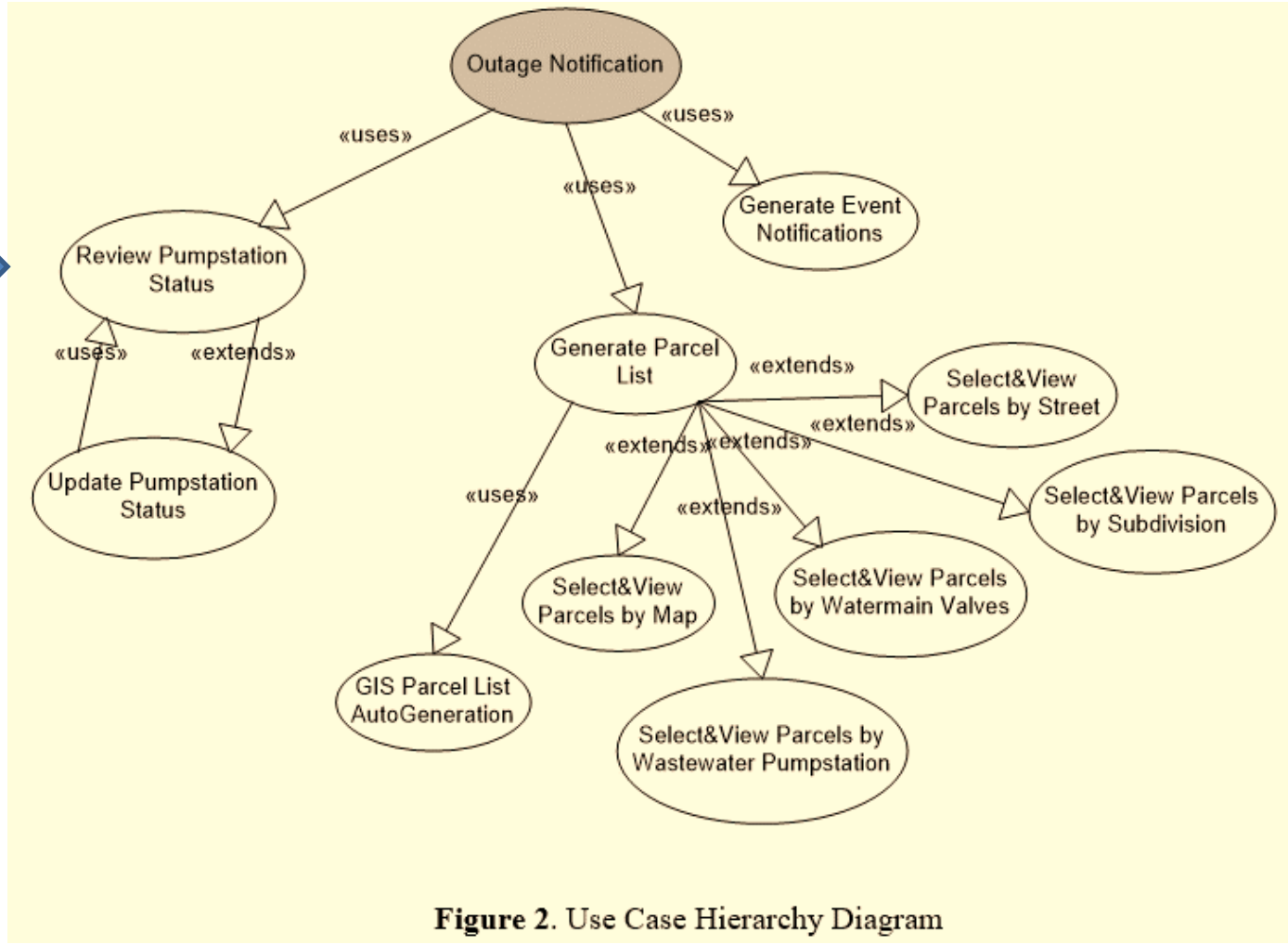


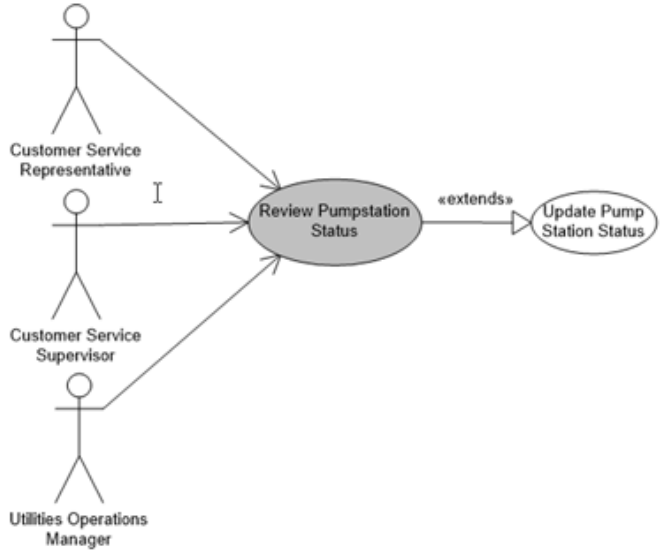
Figure 2. Use Case Hierarchy Diagram

Validation

Did they build the right application?

Each bubble represents a functional capability ("use case") of the application

Functional model



Validation

- *Did they build the right application?*
- *Does it do what the organization needs?*

Additional software requirements for handling a security failure in the context of the use case:

Contingency to Normal Operations:	Fail Case		Consequence to Failure
Security Requirements:			
Secure Requirements:			
Security Constraints:			
Data Collection & Privacy:	Confidentiality	Integrity	Availability
Associated Risks:			

Use Case ID:	1		
Use Case Name:	**Review Pumpstation Status**		
Iteration:	Focused		
Created By:	[redacted]	Last Updated By:	David Lanter
Date Created:	6-17-2005	Date Last Updated:	7-6-2005
Actor:	Customer Service Representative (CSR) Customer Service Supervisor (CSS) Utilities Operations Manager (UOM)		
Description:	The user (CSR, CSS or UOM) confirms that the pump stations' statuses are up-to-date, before generating an outage event notification list.		
Triggers:	Outage event has occurred or is planned.		
Preconditions:	• Up to date pump station GIS feature class dataset with current pump station status values exist are presented to user within GIS application's map user interface. • Parcel GIS feature class dataset must exist and presented to user within GIS application's map user interface. • GIS Data Server online • GIS Web Server online		
Postconditions:	None		
Priority:	Unknown		
Frequency of Use:	Moderate		
Normal Course of Events:	1. User receives information that an outage has occurred, or is planned. 2. User invokes the GIS Outage Notification application. 3. User reviews display of pump stations' statuses on GIS' application's map. 4. User confirms that the pump stations' statuses are up-to-date in the GIS.		
Alternative Courses:	3a. User reviews display of pump station's statuses in pump station status list		
Exceptions:	If the CSR or CSS determines that the pump stations' statuses are not up-to-date, they will notify the UOM responsible for updating the pump station statuses.		
Extensions:	Use Case 2 – Update Pumpstation Status		
Includes:	None		
Related Business Rules:	None		
Special Requirements:	None		
Assumptions:	User provided with GUI control to invoke this use case.		
Notes and Issues:	• It is not clear how User knows for certain that the pump stations' statuses are correct in the GIS. • SCADA or a real-time data feedback system is required to assure that pump stations' statuses are all correct and up-to-date. • CSR or CSS must work through the UOM to assure that the status of the pumpstations are correct.		

Contingency to Normal Operations Outline effects of a failure to the system. This includes:

- ***Fail Case*** — what to do when things go wrong
- ***Consequences of Failure*** — the negative business affects when a security incident occurs

Security Requirements Outline how the attack surfaces are being protected from external attackers and how inherent vulnerabilities will be mitigated, accepted, or avoided

Secure Requirements How does this use case address overall security of the system(s) involved, business processes, and individual business units

Security Constraints What constraints does this use case put on the security of the system and/or processes by limiting capabilities of security software, hardware, and/or procedures?

Data Collection & Privacy What are the impacts of breaches to Confidentiality, Integrity, and Availability of the process, data being collected, and the privacy of the overall system?

Associated Risks What are the security specific risks that come along with running this use case?

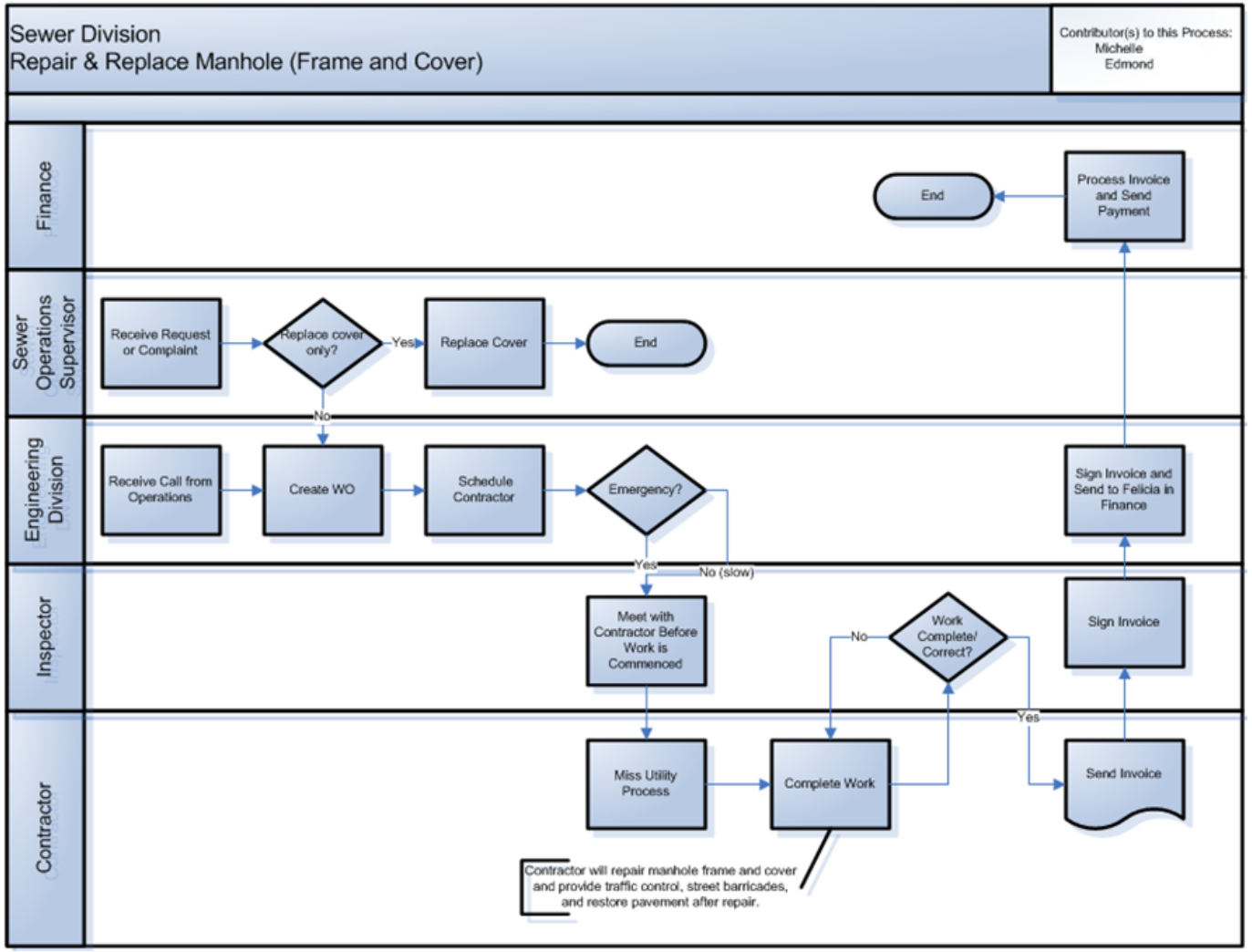
Behavioral models – “swim lane” model

Validation

“Did they build the right application?”

Verification

“Did they build the application the way the organization functions and needs it to work?”



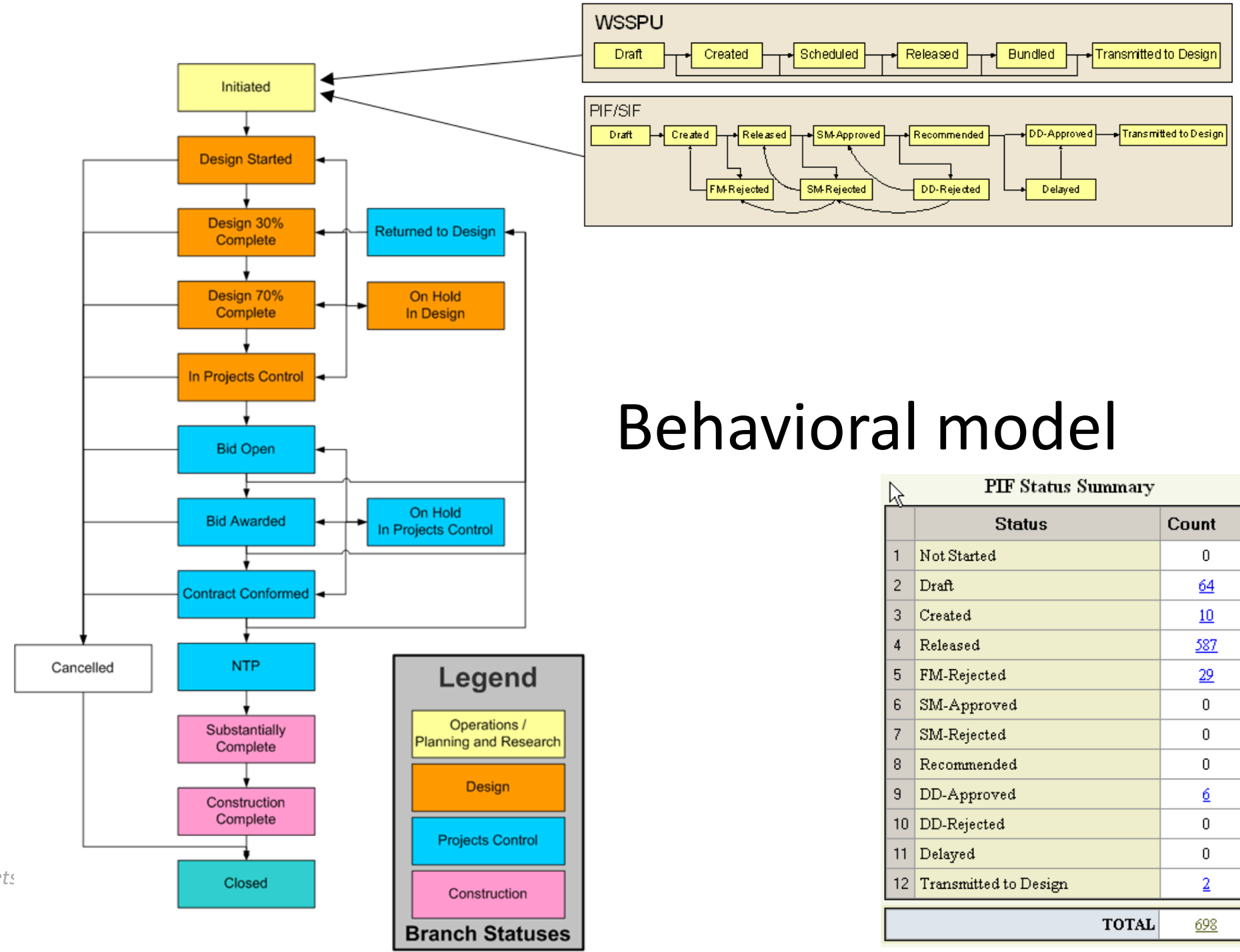
3. Behavioral model

Validation

"Did they build the right application?"

Verification

"Did they build the application right?"



Behavioral model

PIF Status Summary

	Status	Count
1	Not Started	0
2	Draft	64
3	Created	10
4	Released	587
5	FM-Rejected	29
6	SM-Approved	0
7	SM-Rejected	0
8	Recommended	0
9	DD-Approved	6
10	DD-Rejected	0
11	Delayed	0
12	Transmitted to Design	2
	TOTAL	698

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Design

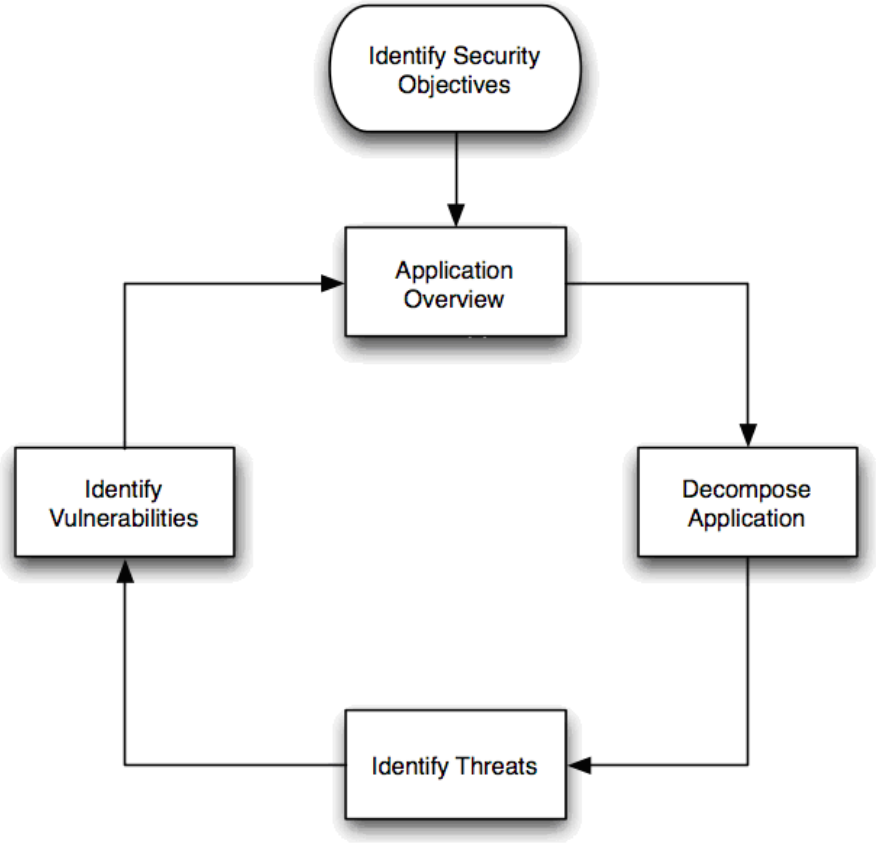
- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

+ Threat modeling, + Attack surface analysis,…

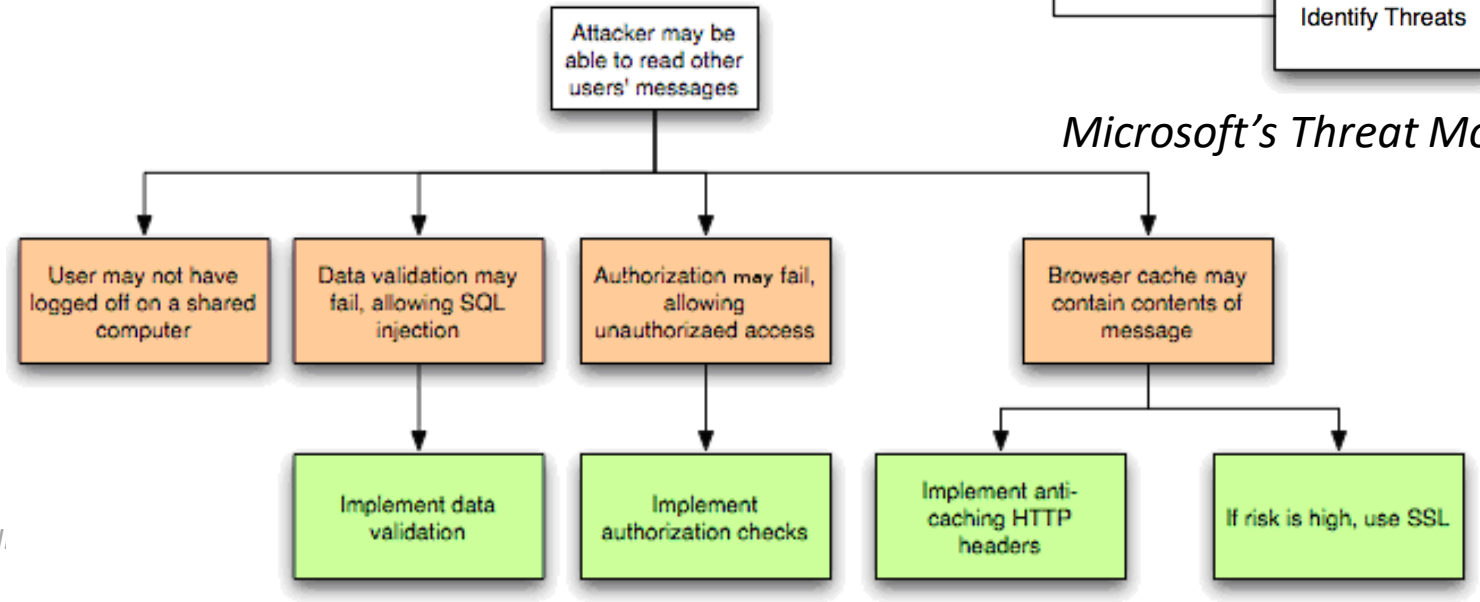
SDLC Design Security

Threat modeling *is a systematic approach for understanding how different threats could be realized and a successful attack could take place*

…leading to mitigations

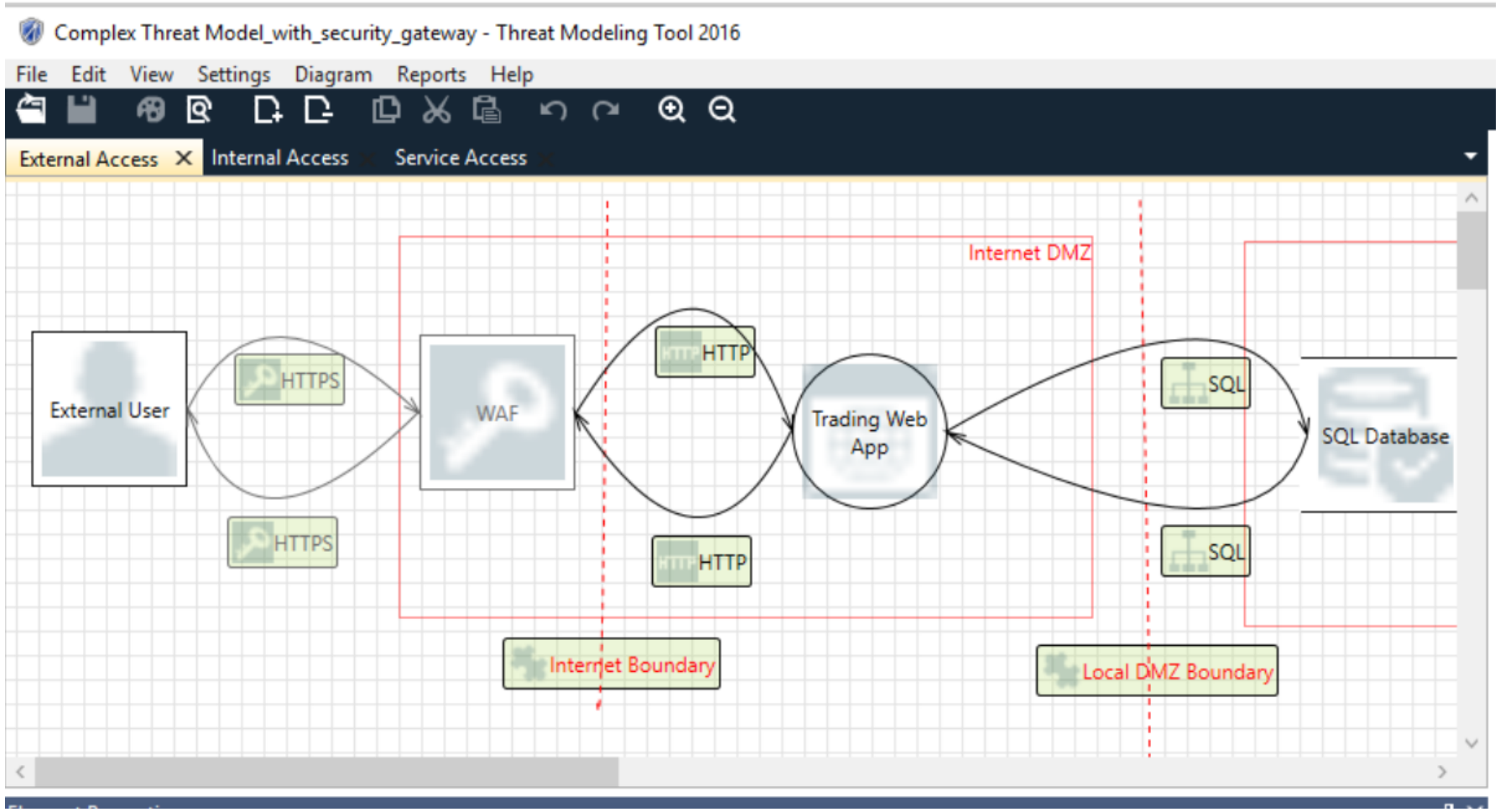


Microsoft's Threat Modeling Process



SDLC Design Security

[Microsoft's Threat Modeling Tool]

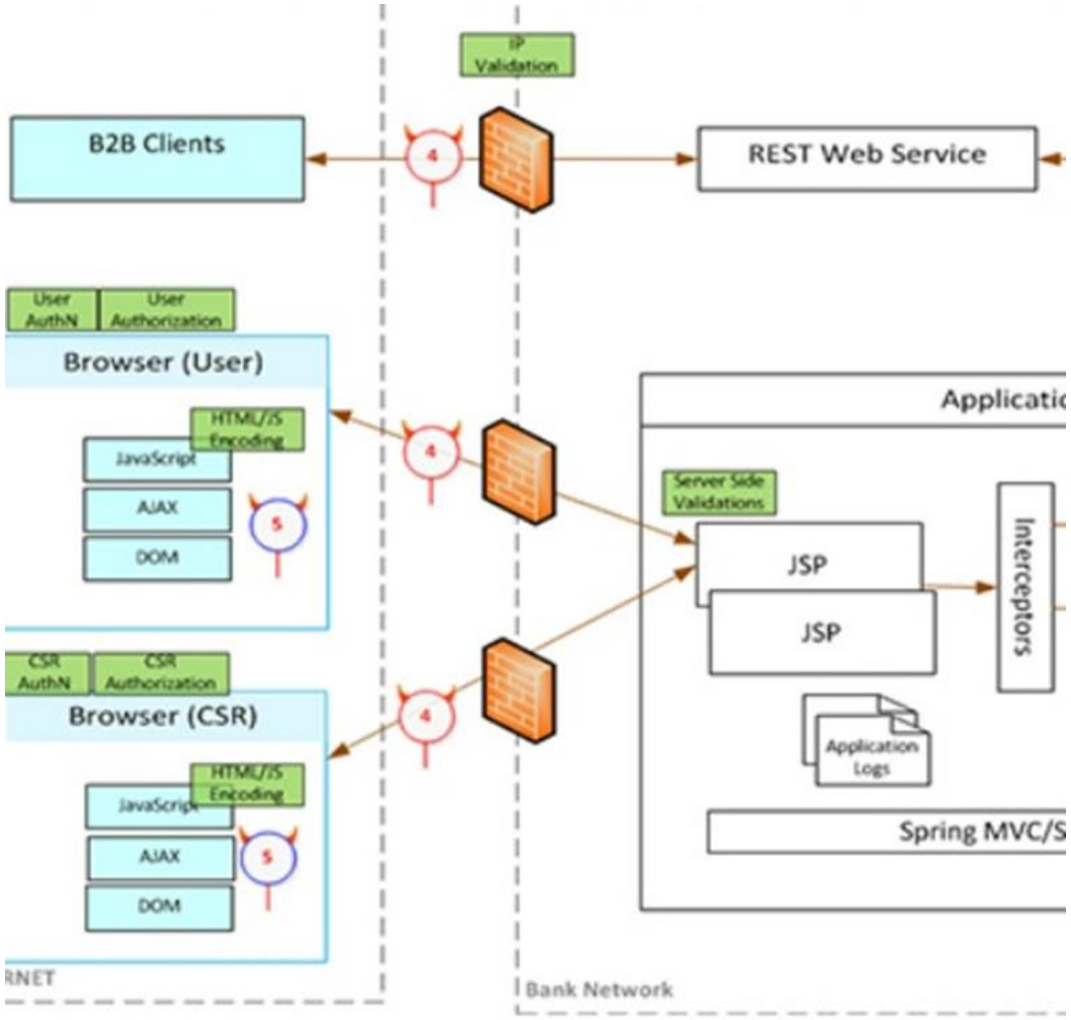


SDLC Design Security

Attack surface *is what is available to be used by an attacker against the application itself*

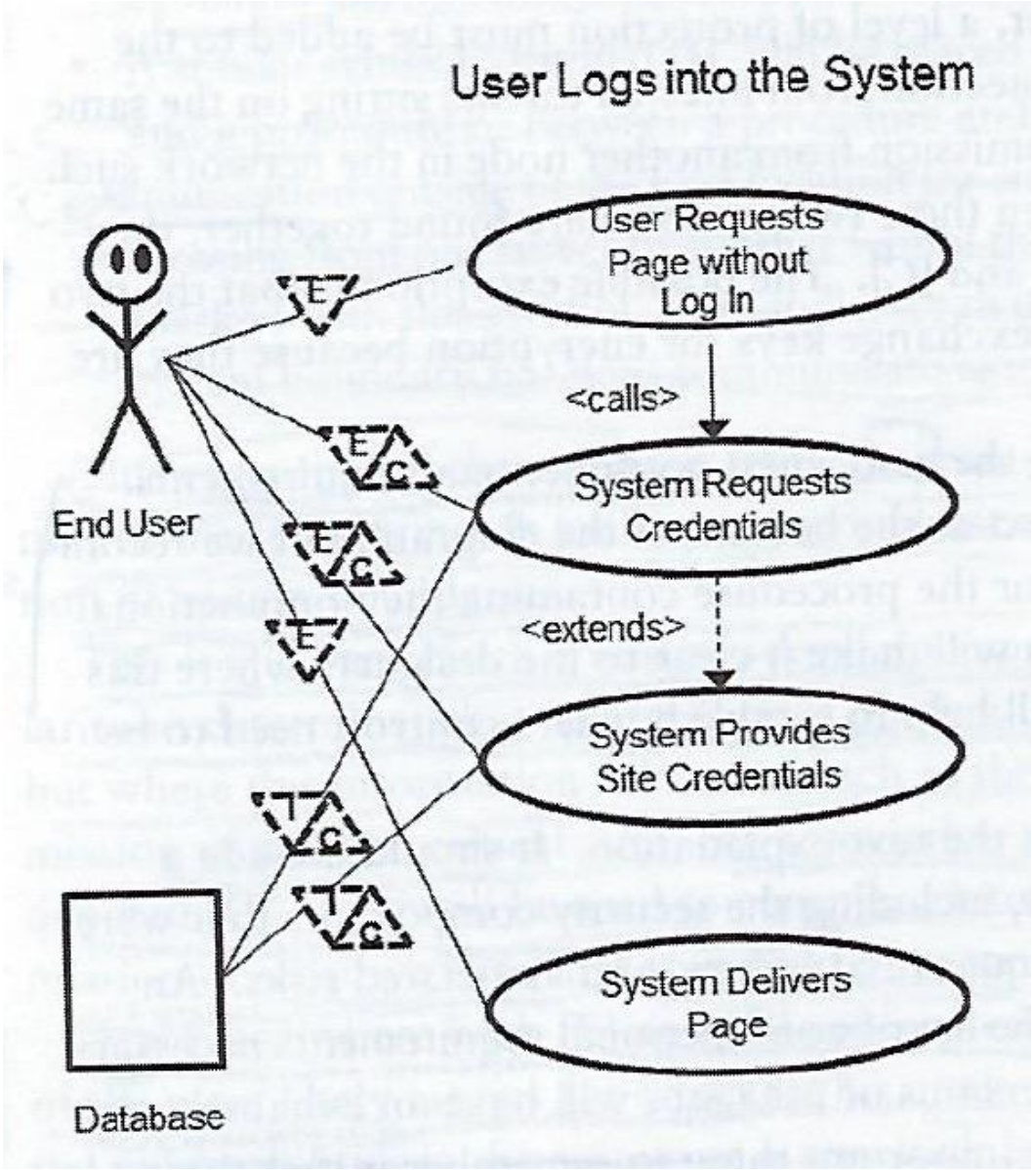
Goal of attack surface analysis is to identify and reduce the amount of code and functionality accessible to untrusted users

Development team should reduce the attack surface as much as possible to remove "resources" that can be used as avenues for the attacker to use



An example use case with notations for communication and transfer of sensitive information across system boundaries

E = External access
E/C = External data communication
I/C = Internal Communication



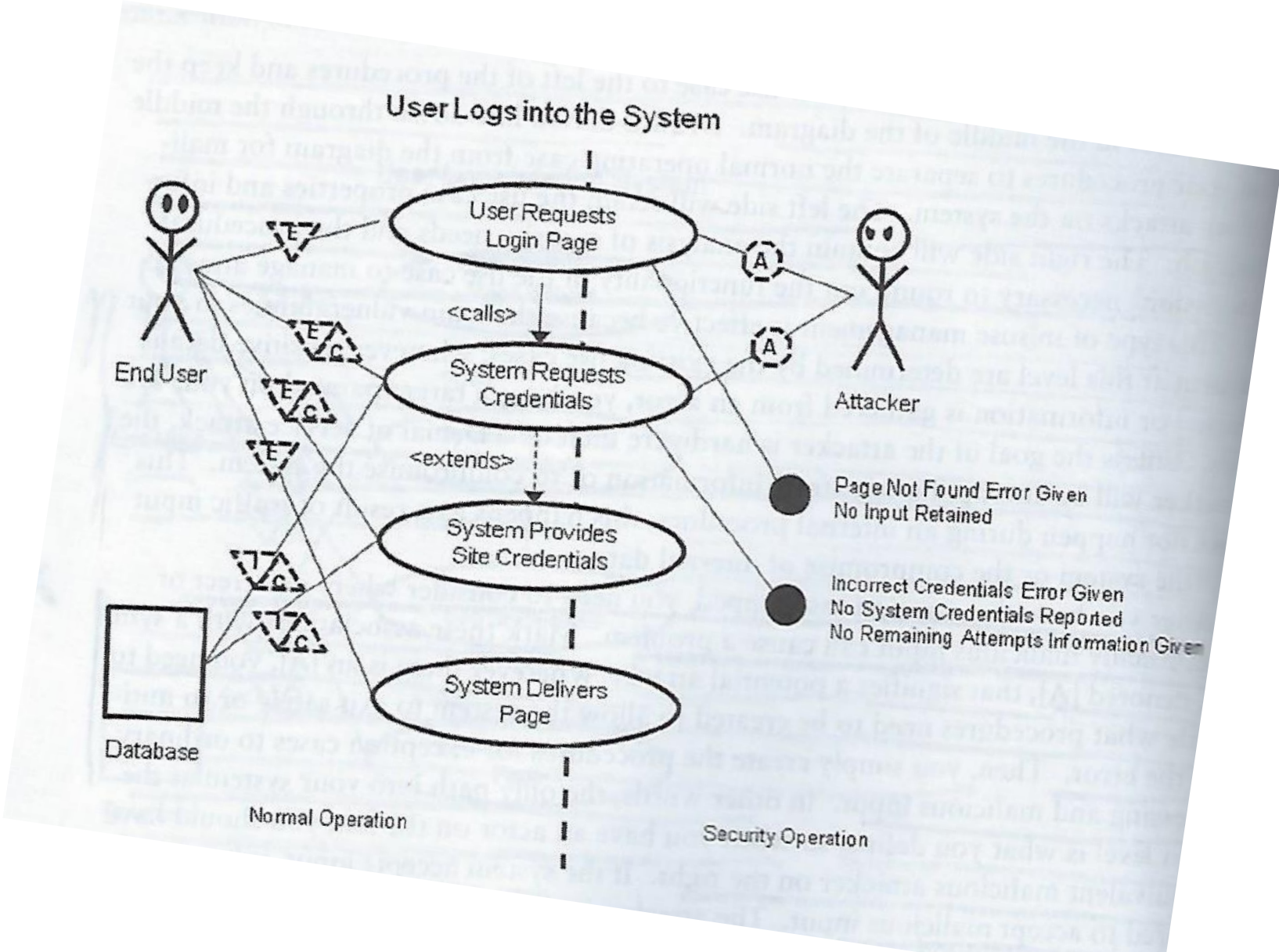
Richardson, T. and Thies, C. (2013) Secure Software Design

An example of the "Misuse Management Method" identifying possible attack points for each activity, and the "fail" use case state for each

E = External access
E/C = External data communication
I/C = Internal Communication

A = Attack

- = Fail use case



Richardson, T. and Thies, C. (2013) Secure Software Design

https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html

Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

Home > CWE Top 25 > 2021

Home | About | CWE List | Scoring | Mapping Guidance | Community | News | Search

2021 CWE Top 25 Most Dangerous Software Weaknesses

Top 25 | Analysis | Methodology | Scoring Metrics | On the Cusp | Limitations | Remapping | Ongoing Improvement

List of the most widespread and critical weaknesses that can lead to serious vulnerabilities in software. These weaknesses are often easy to find and exploit..

They are dangerous because they allow adversaries to completely take over execution of software, steal data, or prevent the software from working

Rank	ID	Name	Score
[1]	CWE-787	Out-of-bounds Write	65.93
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84
[3]	CWE-125	Out-of-bounds Read	24.9
[4]	CWE-20	Improper Input Validation	20.47
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54
[7]	CWE-416	Use After Free	16.83
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45
[11]	CWE-306	Missing Authentication for Critical Function	7.93
[12]	CWE-190	Integer Overflow or Wraparound	7.12
[13]	CWE-502	Deserialization of Untrusted Data	6.71
[14]	CWE-287	Improper Authentication	6.58
[15]	CWE-476	NULL Pointer Dereference	6.54
[16]	CWE-798	Use of Hard-coded Credentials	6.27
[17]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	5.84
[18]	CWE-862	Missing Authorization	5.47
[19]	CWE-276	Incorrect Default Permissions	5.09
[20]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	4.74
[21]	CWE-522	Insufficiently Protected Credentials	4.21
[22]	CWE-732	Incorrect Permission Assignment for Critical Resource	4.2
[23]	CWE-611	Improper Restriction of XML External Entity Reference	4.02
[24]	CWE-918	Server-Side Request Forgery (SSRF)	3.78
[25]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	3.58

https://owasp.org/Top10/

Welcome to the OWASP Top 10 - 2021

A01 Broken Access Control
A02 Cryptographic Failures
A03 Injection
A04 Insecure Design
A05 Security Misconfiguration
A06 Vulnerable and Outdated Components
A07 Identification and Authentication Failures
A08 Software and Data Integrity Failures
A09 Security Logging and Monitoring Failures
A10 Server-Side Request Forgery (SSRF)

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

+ Threat modeling, + Attack surface analysis,…

Develop (“*make*”) / Implement (“*buy*”)

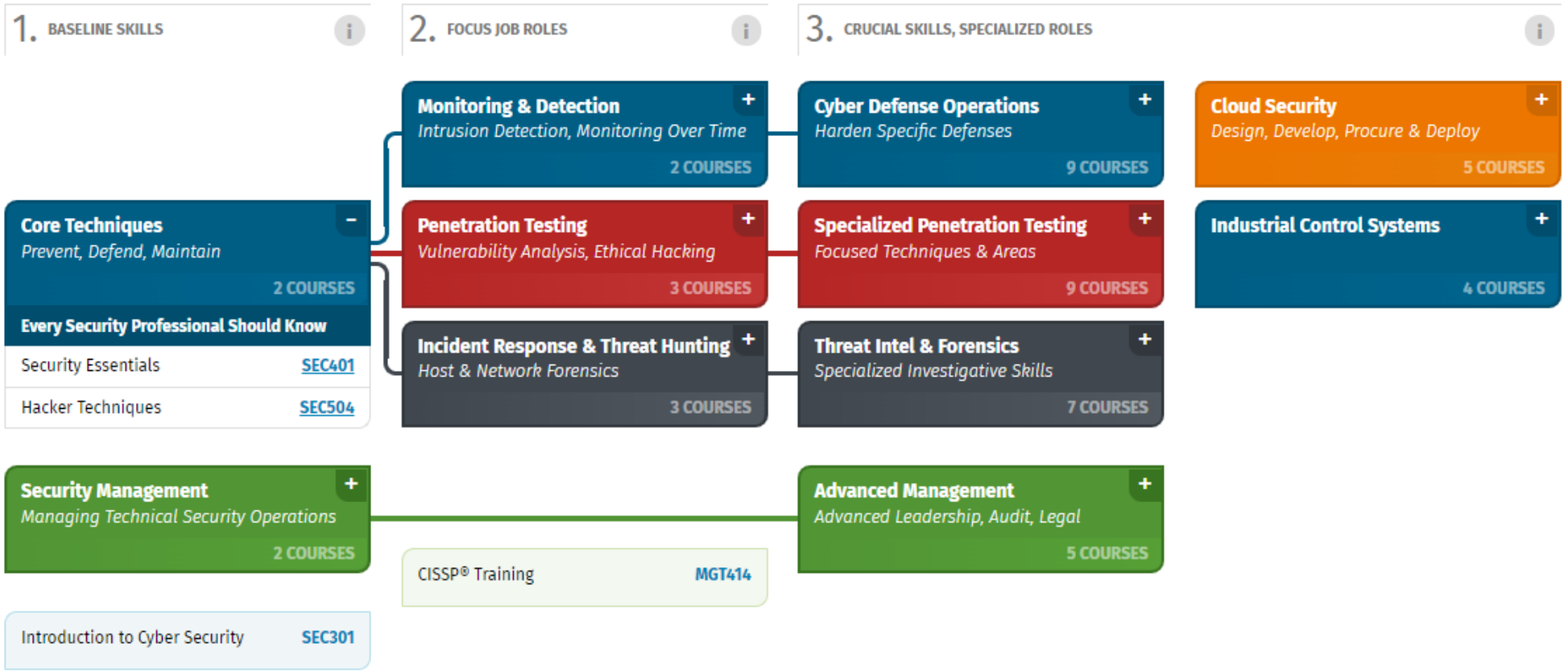
- *Source code control system, code reviews, daily builds, automated CASE tools…*

+ Developer security training, + Secure code repositories + Static analysis, + Software Composition(Component) Analysis +,…

SANS

Train and Certify | Manage Your Team | Resources | Focus Areas | Get Involved | About

Cyber Security Skills Roadmap



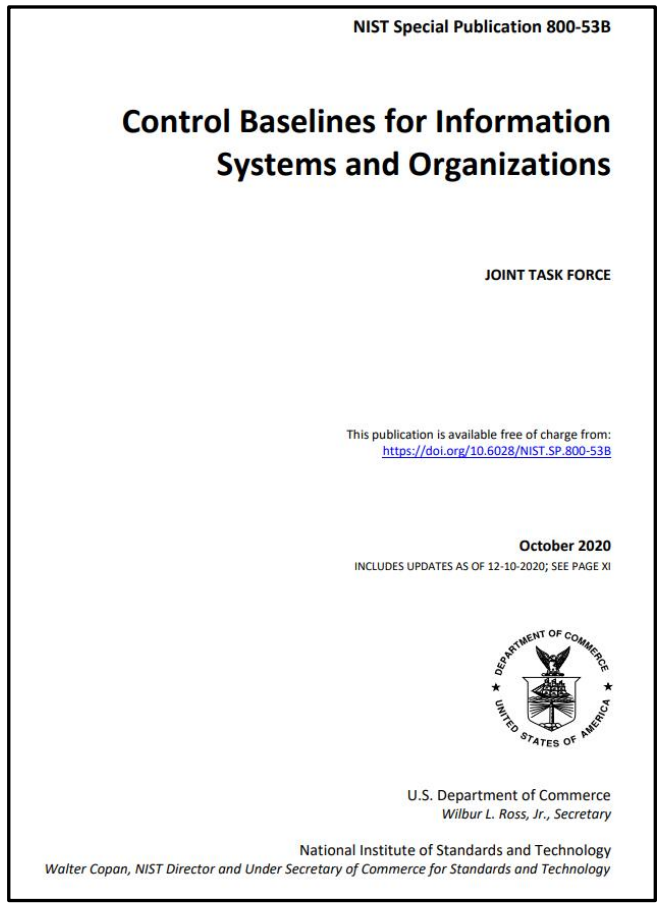
Code Repositories

- **Source Code Control System (SCCS)** is a version control system designed to track changes in source code and other text files during the development of a piece of software

- A **Code Repository** is a term used by most of the different source control tools to refer to the collection of source code

Name	Code review	Bug tracking	Web hosting	Wiki	Translation system	Shell server	Mailing list	Forum	Personal repository	Private repository	Announce	Build system	Team	Release binaries	Self-hosting
Assembla	Yes[22]	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes[23]	Yes	Yes	Yes	Unknown	No
Azure DevOps Services	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Commercially (Azure DevOps Server)
Bitbucket	Yes[24]	Yes[a]	Yes[25]	Yes	No	No	No	No	Yes	Yes[b]	No	Yes[26]	Yes	No[27]	Commercially (Bitbucket Server formerly Stash)[c]
Buddy	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes[d]	Yes	Yes	Yes
CloudForge	Unknown	Yes	Yes	Yes	No	No	No	No	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	No
GForge	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Gitea	Yes	Yes	No	Yes	Unknown	Unknown	Unknown	Unknown	Yes	Yes	Unknown	3rd-party (e.g. Travis CI, Appveyor and others)[28]	Yes	Unknown	Yes
GitHub	Yes[29]	Yes[30][e]	Yes[31]	Yes	No	No	No	No	Yes	Yes[f]	Yes	3rd-party (e.g. Travis CI, Appveyor and others)[32]	Yes	Yes	Commercially (GitHub Enterprise)
GitLab	Yes[33]	Yes	Yes[34]	Yes	No	No	No	No	Yes	Yes	Yes	Yes[35]	Yes	Yes[36]	Yes[g]
GNU Savannah	Yes[37]	Yes	Yes	No	No	Yes	Yes	No[38]	No	No	Yes	No	Yes	Unknown	Yes
Helix TeamHub	Yes[39]	Yes	No	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes, with hooks. Jenkins, TeamCity, etc.	No	Yes	Yes
java.net/Project Kenai	Unknown	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Unknown	No
Kallithea	Yes	No	Yes	No	No	Unknown	No	No	Yes	Yes	No	No	Yes	Yes	Yes
Launchpad	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes[h]	Yes	Yes[i]	Yes	Unknown	Yes
OSDN	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No
Ourproject.org	Unknown	Yes	Yes	Yes	No	Unknown	Yes	Yes	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Yes
Phabricator	Yes	Yes	Yes	Yes	Unknown	Yes	Unknown	Yes	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Yes
RhodeCode	Yes	No	Yes	No	No	Unknown	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes
SourceForge	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes[j]	Yes	No	Yes	Yes	Yes

Testing/Validation



NIST Special Publication 800-53B

Control Baselines for Information Systems and Organizations

JOINT TASK FORCE

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-53B

October 2020
INCLUDES UPDATES AS OF 12-10-2020; SEE PAGE XI

U.S. Department of Commerce
Wilbur L. Ross, Jr., Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

CNTL NO.	CONTROL NAME	PRIORITY	INITIAL CONTROL BASELINES		
			LOW	MOD	HIGH
System and Services Acquisition					
SA-1	System and Services Acquisition Policy and Procedures	P1	SA-1	SA-1	SA-1
SA-2	Allocation of Resources	P1	SA-2	SA-2	SA-2
SA-3	System Development Life Cycle	P1	SA-3	SA-3	SA-3
SA-4	Acquisition Process	P1	SA-4 (10)	SA-4 (1) (2) (9) (10)	SA-4 (1) (2) (9) (10)
SA-5	Information System Documentation	P2	SA-5	SA-5	SA-5
SA-8	Security Engineering Principles	P1	Not Selected	SA-8	SA-8
SA-9	External Information System Services	P1	SA-9	SA-9 (2)	SA-9 (2)
SA-10	Developer Configuration Management	P1	Not Selected	SA-10	SA-10
SA-11	Developer Security Testing and Evaluation	P1	Not Selected	SA-11	SA-11
SA-15	Development Process, Standards, and Tools	P2	Not Selected	Not Selected	SA-15
SA-16	Developer-Provided Training	P2	Not Selected	Not Selected	SA-16
SA-17	Developer Security Architecture and Design	P1	Not Selected	Not Selected	SA-17

NIST Special Publication 800-53A
Revision 4

Assessing Security and Privacy Controls in Federal Information Systems and Organizations

Building Effective Assessment Plans

JOINT TASK FORCE TRANSFORMATION INITIATIVE

This publication is available free of charge from:
http://dx.doi.org/10.6028/NIST.SP.800-53Ar4

December 2014
INCLUDES UPDATES AS OF 12-18-2014

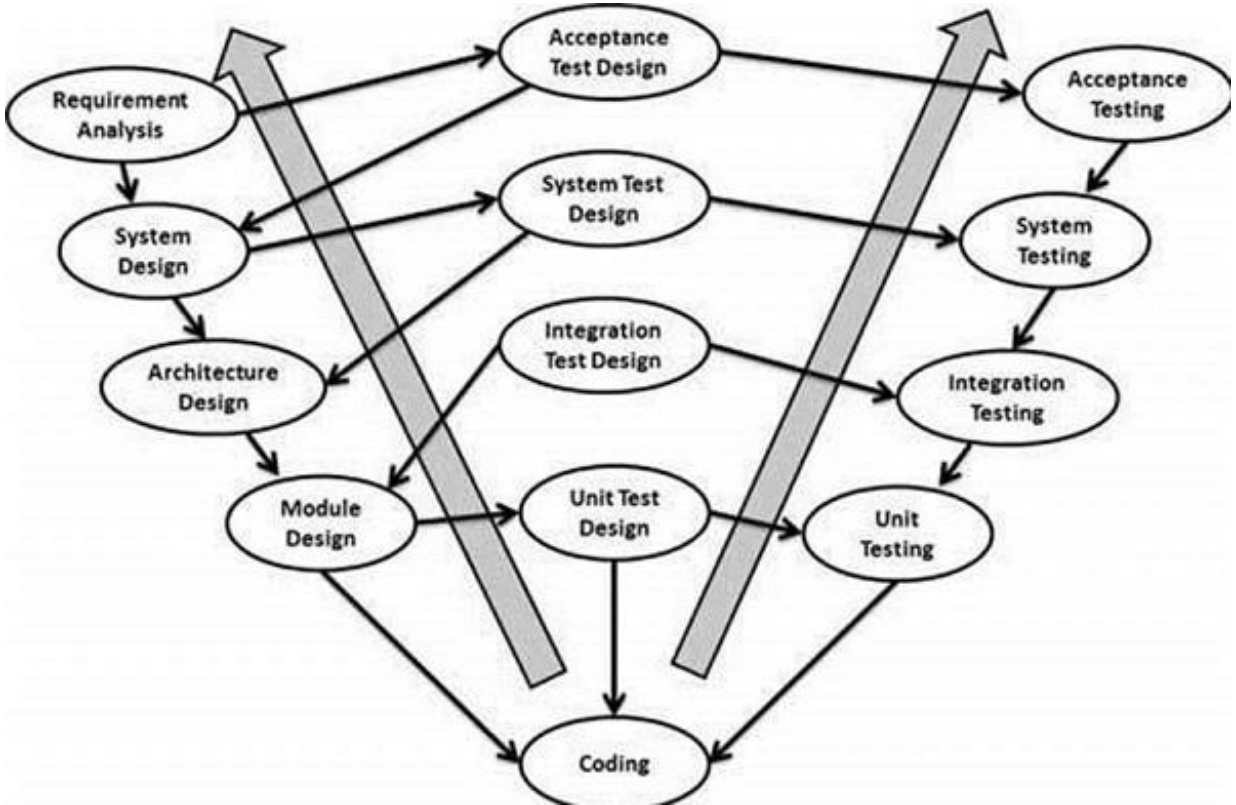
U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Acting Under Secretary of Commerce for Standards and Technology and Acting Director

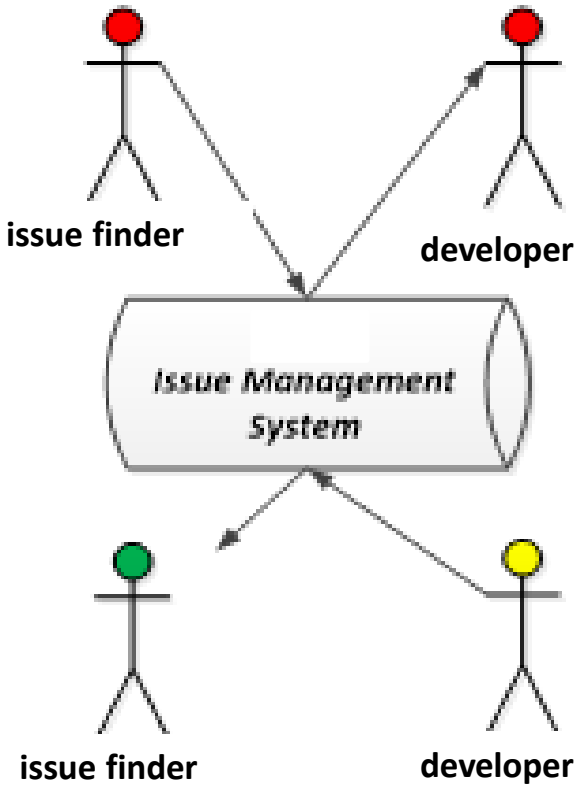
SA-11	DEVELOPER SECURITY TESTING AND EVALUATION			
	ASSESSMENT OBJECTIVE: *Determine if the organization:*			
	SA-11(a)	*requires the developer of the information system, system component, or information system service to create and implement a security plan;*		
	SA-11(b)	**SA-11(b)[1]**	*defines the depth of testing/evaluation to be performed by the developer of the information system, system component, or information system service;*	
		SA-11(b)[2]	*defines the coverage of testing/evaluation to be performed by the developer of the information system, system component, or information system service;*	
		SA-11(b)[3]	*requires the developer of the information system, system component, or information system service to perform one or more of the following testing/evaluation at the organization-defined depth and coverage:*	
			SA-11(b)[3][a]	*unit testing/evaluation;*
			SA-11(b)[3][b]	*integration testing/evaluation;*
			SA-11(b)[3][c]	*system testing/evaluation; and/or*
			SA-11(b)[3][d]	*regression testing/evaluation;*
	SA-11(c)	*requires the developer of the information system, system component, or information system service to produce evidence of:*		
		SA-11(c)[1]	*the execution of the security assessment plan;*	
		SA-11(c)[2]	*the results of the security testing/evaluation;*	
	SA-11(d)	*requires the developer of the information system, system component, or information system service to implement a verifiable flaw remediation process; and*		
	SA-11(e)	*requires the developer of the information system, system component, or information system service to correct flaws identified during security testing/evaluation.*		
	POTENTIAL ASSESSMENT METHODS AND OBJECTS: **Examine**: [*SELECT FROM:* System and services acquisition policy; procedures addressing system developer security testing; procedures addressing flaw remediation; solicitation documentation; acquisition documentation; service-level agreements; acquisition contracts for the information system, system component, or information system service; system developer security test plans; records of developer security testing results for the information system, system component, or information system service; security flaw and remediation tracking records; other relevant documents or records]. **Interview**: [*SELECT FROM:* Organizational personnel with system and services acquisition responsibilities; organizational personnel with information security responsibilities; organizational personnel with developer security testing responsibilities; system developers]. **Test**: [*SELECT FROM:* Organizational processes for monitoring developer security testing and evaluation; automated mechanisms supporting and/or implementing the monitoring of developer security testing and evaluation].			

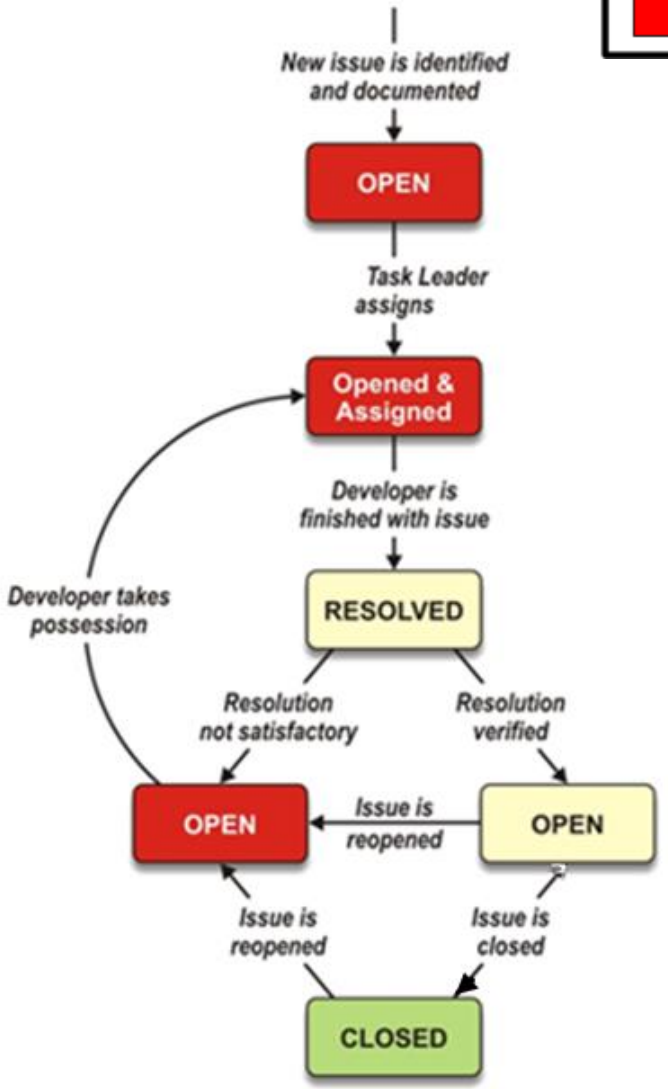
Software Application Testing

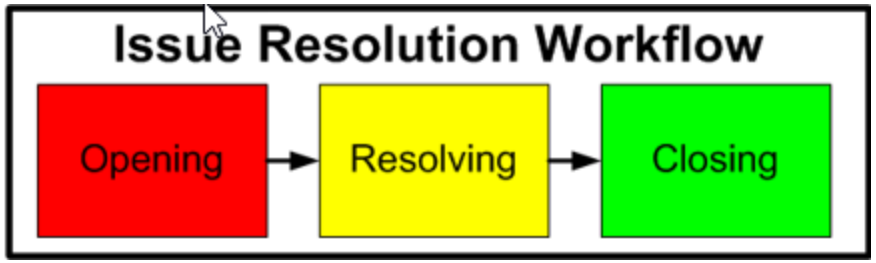
- A test plan is developed during the analysis phase
- During the design phase, unit, system and integration test plans are developed
- The actual testing is done during implementation
- Written test plans provide improved communication among all parties involved in testing

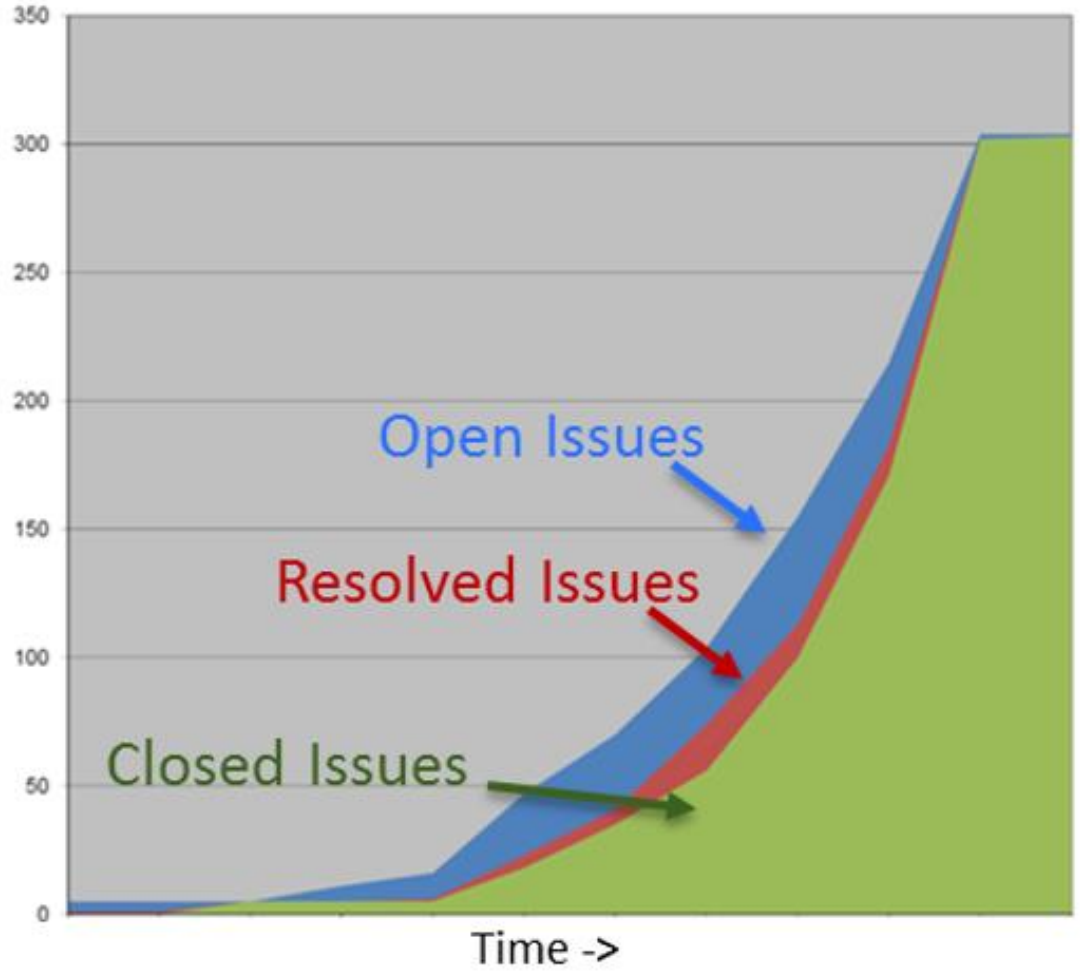


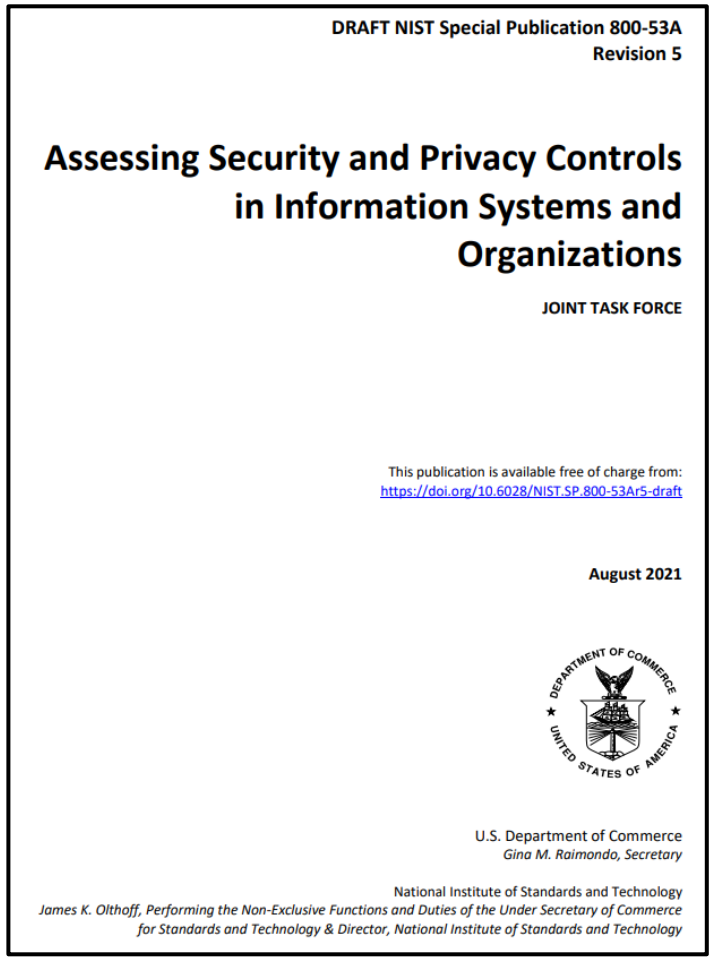
Testing/validation











DRAFT NIST Special Publication 800-53A
Revision 5

Assessing Security and Privacy Controls in Information Systems and Organizations

JOINT TASK FORCE

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-53Ar5-draft

August 2021

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology

SA-11(01)	DEVELOPER TESTING AND EVALUATION | STATIC CODE ANALYSIS	
	ASSESSMENT OBJECTIVE: *Determine if:*	
	SA-11(01)[01]	the developer of the system, system component, or system service is required to employ static code analysis tools to identify common flaws;
	SA-11(01)[02]	the developer of the system, system component, or system service is required to employ static code analysis tools to document the results of the analysis.

Application Security Testing (AST)

"**Static AST (SAST)** technology analyzes an application's source, bytecode or binary code for security vulnerabilities typically at the programming and/or testing software life cycle (SLC) phases." *White-box testing*

"**Software composition analysis (SCA)** technology is used to identify open-source and third-party components in use in an application, and their known security vulnerabilities."

Gardner, D., Horvath, M., Zumerle, D. (2021), "Magic Quadrant for Application Security Testing", Gartner

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

+ Threat modeling, + Attack surface analysis,…

Develop (“*make*”) / Implement (“*buy*”)

- *Source code control system, code reviews, daily builds, automated CASE tools…*

+ Developer security training, + Static analysis, + Secure code repositories,…

Testing/Validation

- *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*

+ Dynamic analysis, + Fuzzing,…

Application Security Testing (AST)

"**Static AST (SAST)** technology analyzes an application's source, bytecode or binary code for security vulnerabilities typically at the programming and/or testing software life cycle (SLC) phases."

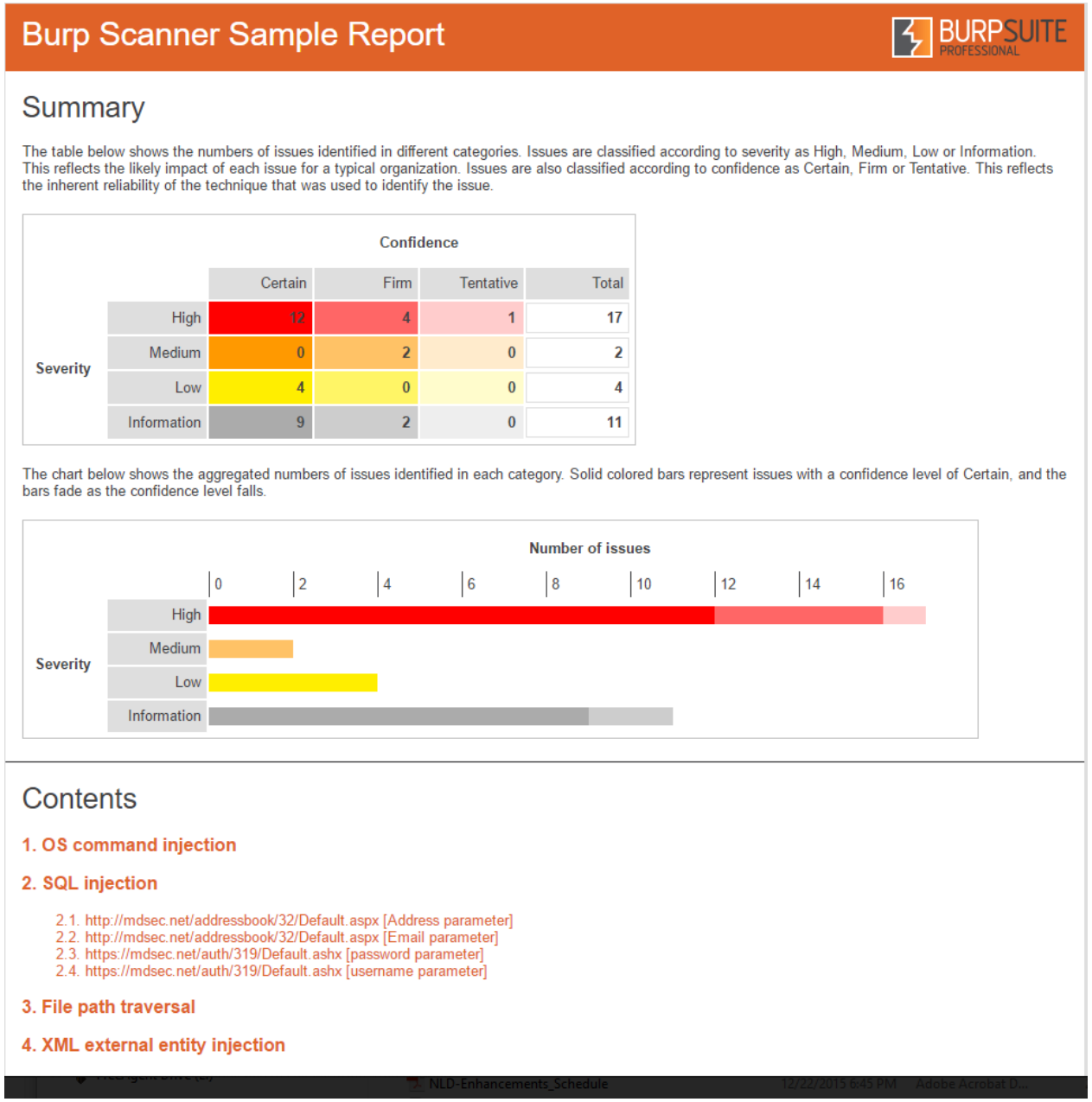
White-box testing

"**Software composition analysis (SCA)** technology is used to identify open-source and third-party components in use in an application, and their known security vulnerabilities."

"**Dynamic AST (DAST)** technology analyzes applications in their dynamic, running state during testing or operational phases. DAST simulates attacks against an application (typically web-enabled applications and services), analyzes the application's reactions and, thus, determines whether it is vulnerable."

Black-box testing

Gardner, D., Horvath, M., Zumerle, D. (2021), "Magic Quadrant for Application Security Testing", Gartner



Burp Scanner Sample Report

BURPSUITE PROFESSIONAL

Summary

The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low or Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

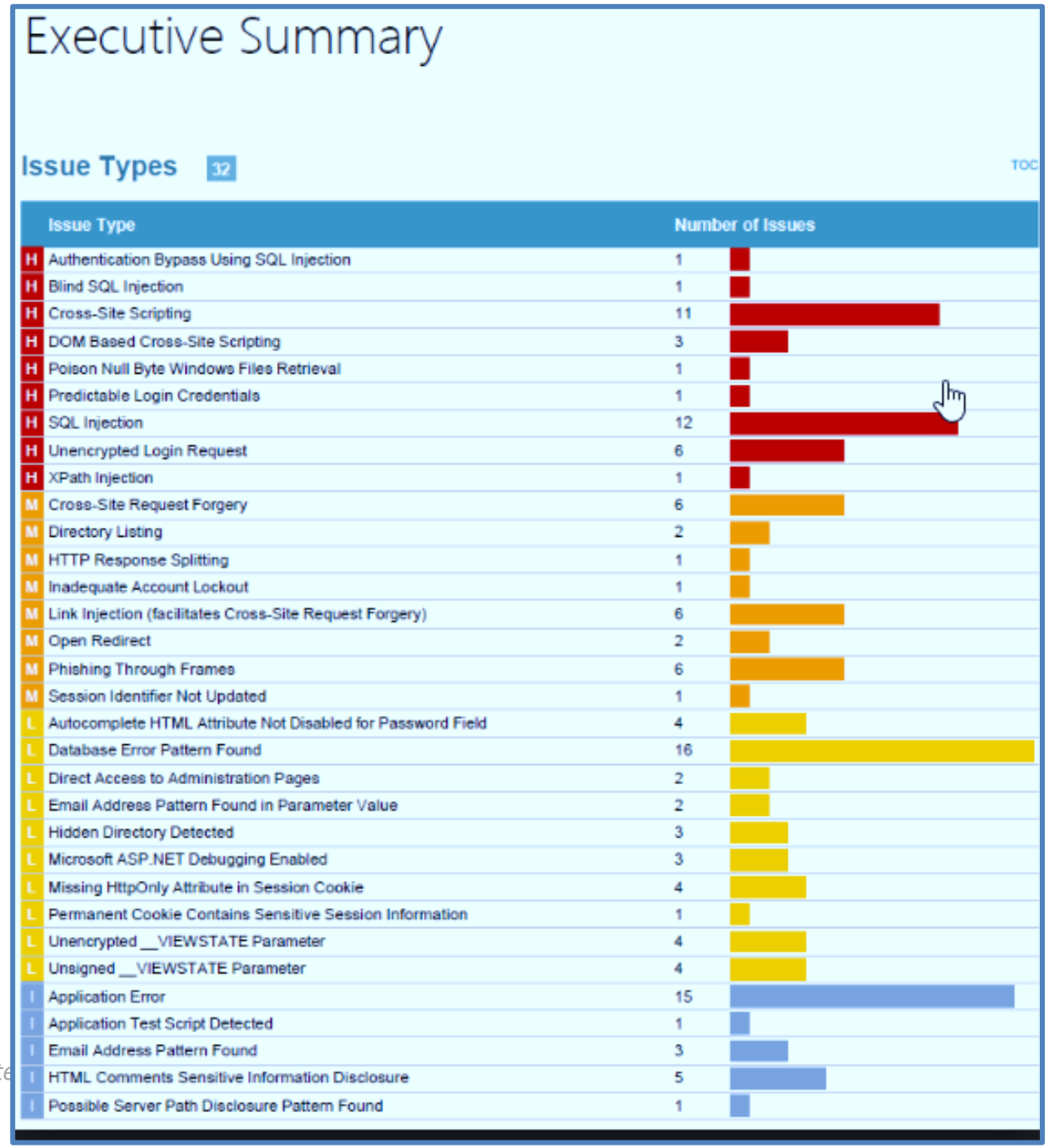
Severity	Confidence			
	Certain	Firm	Tentative	Total
High	12	4	1	17
Medium	0	2	0	2
Low	4	0	0	4
Information	9	2	0	11

The chart below shows the aggregated numbers of issues identified in each category. Solid colored bars represent issues with a confidence level of Certain, and the bars fade as the confidence level falls.

Contents

Application Security Testing result reports

- Applications should not accepted until *all high and medium issues resolved!*



Executive Summary

Issue Types 32

TOC

	Issue Type	Number of Issues
H	Authentication Bypass Using SQL Injection	1
H	Blind SQL Injection	1
H	Cross-Site Scripting	11
H	DOM Based Cross-Site Scripting	3
H	Poison Null Byte Windows Files Retrieval	1
H	Predictable Login Credentials	1
H	SQL Injection	12
H	Unencrypted Login Request	6
H	XPath Injection	1
M	Cross-Site Request Forgery	6
M	Directory Listing	2
M	HTTP Response Splitting	1
M	Inadequate Account Lockout	1
M	Link Injection (facilitates Cross-Site Request Forgery)	6
M	Open Redirect	2
M	Phishing Through Frames	6
M	Session Identifier Not Updated	1
L	Autocomplete HTML Attribute Not Disabled for Password Field	4
L	Database Error Pattern Found	16
L	Direct Access to Administration Pages	2
L	Email Address Pattern Found in Parameter Value	2
L	Hidden Directory Detected	3
L	Microsoft ASP.NET Debugging Enabled	3
L	Missing HttpOnly Attribute in Session Cookie	4
L	Permanent Cookie Contains Sensitive Session Information	1
L	Unencrypted __VIEWSTATE Parameter	4
L	Unsigned __VIEWSTATE Parameter	4
I	Application Error	15
I	Application Test Script Detected	1
I	Email Address Pattern Found	3
I	HTML Comments Sensitive Information Disclosure	5
I	Possible Server Path Disclosure Pattern Found	1

Application Security Testing result reports

- Applications should not accepted until *all high and medium issues resolved!*

Automated application security testing tools often provide vulnerability reports

Web Application Report

This report includes important security information about your web application.

Security Report

This report was created by IBM Security AppScan Standard 8.8.0.0, Rules: 1696
Scan started: 9/8/2016 9:29:09 AM

This report contains the results of a web application security scan performed by IBM Security AppScan Standard.

High severity issues:	79
Medium severity issues:	198
Total security issues included in the report:	277
Total security issues discovered in the scan:	308

Application Security Assessment and Fix Recommendations

Issue Types 21

TOC

	Issue Type	Number of Issues
H	Authentication Bypass Using HTTP Verb Tampering	3
H	Cross-Site Request Forgery	23
H	Cross-Site Scripting	2
H	Microsoft FrontPage Extensions Site Defacement	3
H	Missing Secure Attribute in Encrypted Session (SSL) Cookie	5
H	RC4 cipher suites were detected	1
M	Alternate Version of File Detected	45
M	Body Parameters Accepted in Query	9
M	Browser Exploit Against SSL/TLS (a.k.a. BEAST)	1
M	Cacheable SSL Page Found	67
M	Direct Access to Administration Pages	1
M	Drupal "keys" Path Disclosure	1
M	Insecure "OPTIONS" HTTP Method Enabled	1
M	Microsoft FrontPage Server Extensions Vital Information Leakage	2
M	Microsoft IIS Missing Host Header Information Leakage	1
M	Missing "Content-Security-Policy" header	5
M	Missing Cross-Frame Scripting Defence	4
M	Query Parameter in SSL Request	185
M	Temporary File Download	3
M	Unencrypted __VIEWSTATE Parameter	20
M	Web Application Source Code Disclosure Pattern Found	1

Fix Recommendations 19

TOC

	Remediation Task	Number of Issues
H	Review possible solutions for hazardous character injection	2
M	Add the 'Secure' attribute to all sensitive cookies	5
M	Change server's supported ciphersuites	2
M	Configure your server to allow only required HTTP methods	3
M	Set proper permissions to the FrontPage extension files	3
M	Validate the value of the "Referer" header, and use a one-time-nonce for each submitted form	23
L	Always use SSL and POST (body) parameters when sending sensitive information.	185
L	Apply configuration changes according to Q218180	1
L	Apply proper authorization to administration scripts	1
L	Config your server to use the "Content-Security-Policy" header	5
L	Config your server to use the "X-Frame-Options" header	4
L	Contact the vendor of your product to see if a patch or a fix has been made available recently	1
L	Disable WebDAV, or disallow unneeded HTTP methods	1
L	Do not accept body parameters that are sent in the query string	9
L	Modify FrontPage extension file permissions to avoid information leakage	2
L	Modify your Web.Config file to encrypt the VIEWSTATE parameter	20
L	Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.	67
L	Remove old versions of files from the virtual directory	48
L	Remove source code files from your web-server and apply any relevant patches	1

This report contains the results of a web application security scan performed by IBM Security AppScan Standard.

High severity issues:	79
Medium severity issues:	198
Total security issues included in the report:	277
Total security issues discovered in the scan:	308

Application Security Vulnerability Assessment Report

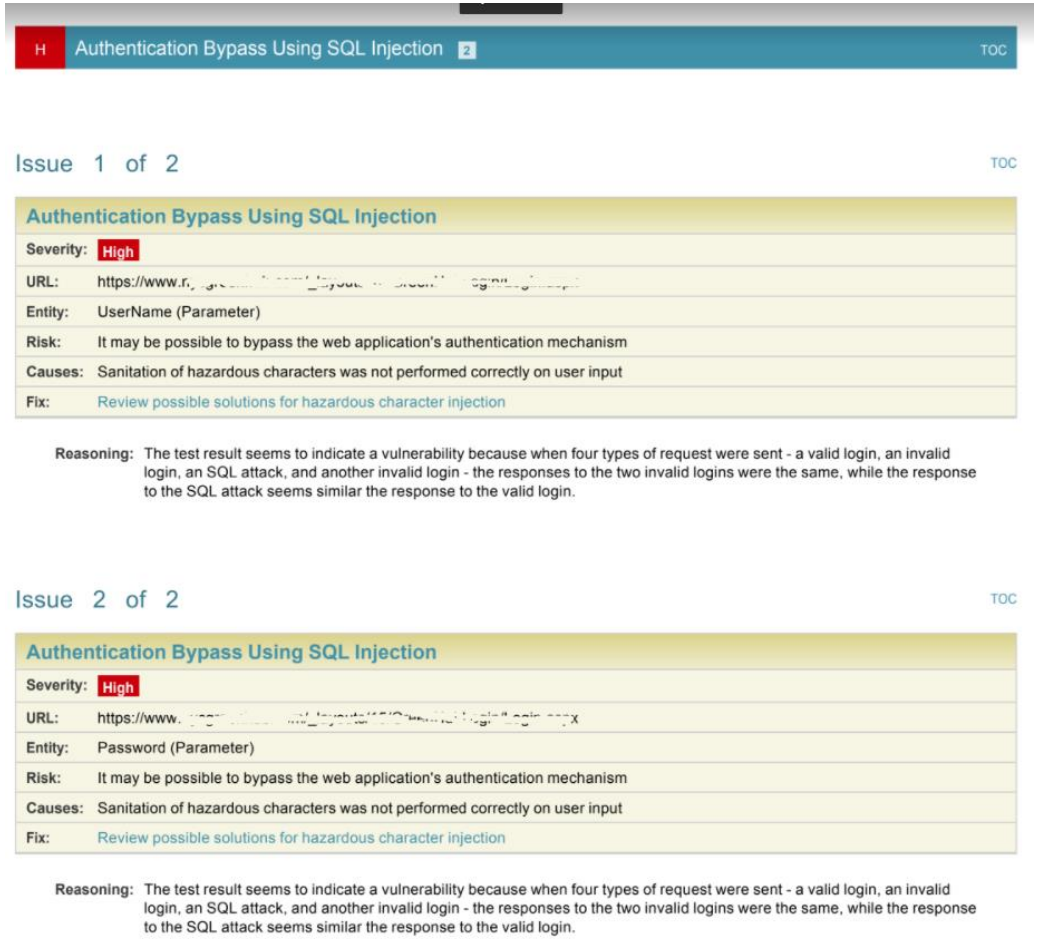
Issues Sorted by Issue Type

- Authentication Bypass Using SQL Injection 2
- Blind SQL Injection 4
- Cross-Site Request Forgery 24
- Cross-Site Scripting 3
- HTTP PUT Method Site Defacement 20
- Inadequate Account Lockout 1
- Microsoft FrontPage Extensions Site Defacement 3
- Missing Secure Attribute in Encrypted Session (SSL) Cookie 1
- Phishing Through URL Redirection 1
- WebDAV MKCOL Method Site Defacement 20
- Alternate Version of File Detected 50
- Cacheable SSL Page Found 26
- Hidden Directory Detected 7
- Microsoft FrontPage Configuration Information Leakage 1
- Microsoft FrontPage Server Extensions Vital Information Leakage 2
- Microsoft IIS Missing Host Header Information Leakage 1
- Query Parameter in SSL Request 66
- Temporary File Download 32
- Unencrypted __VIEWSTATE Parameter 11
- Web Application Source Code Disclosure Pattern Found 2

IBM AppScan example

Advisories

- Authentication Bypass Using SQL Injection
- Blind SQL Injection
- Cross-Site Request Forgery
- Cross-Site Scripting
- HTTP PUT Method Site Defacement
- Inadequate Account Lockout
- Microsoft FrontPage Extensions Site Defacement
- Missing Secure Attribute in Encrypted Session (SSL) Cookie
- Phishing Through URL Redirection
- WebDAV MKCOL Method Site Defacement
- Alternate Version of File Detected
- Cacheable SSL Page Found
- Hidden Directory Detected
- Microsoft FrontPage Configuration Information Leakage
- Microsoft FrontPage Server Extensions Vital Information Leakage
- Microsoft IIS Missing Host Header Information Leakage
- Query Parameter in SSL Request
- Temporary File Download
- Unencrypted __VIEWSTATE Parameter
- Web Application Source Code Disclosure Pattern Found



H Authentication Bypass Using SQL Injection 2 TOC

Issue 1 of 2 TOC

Authentication Bypass Using SQL Injection	
Severity:	High
URL:	https://www.r...
Entity:	UserName (Parameter)
Risk:	It may be possible to bypass the web application's authentication mechanism
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

Reasoning: The test result seems to indicate a vulnerability because when four types of request were sent - a valid login, an invalid login, an SQL attack, and another invalid login - the responses to the two invalid logins were the same, while the response to the SQL attack seems similar the response to the valid login.

Issue 2 of 2 TOC

Authentication Bypass Using SQL Injection	
Severity:	High
URL:	https://www....
Entity:	Password (Parameter)
Risk:	It may be possible to bypass the web application's authentication mechanism
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

Reasoning: The test result seems to indicate a vulnerability because when four types of request were sent - a valid login, an invalid login, an SQL attack, and another invalid login - the responses to the two invalid logins were the same, while the response to the SQL attack seems similar the response to the valid login.

Authentication Bypass Using SQL Injection

TOC

Test Type:

Application-level test

Threat Classification:

Insufficient Authentication

Causes:

Sanitation of hazardous characters was not performed correctly on user input

Security Risks:

It may be possible to bypass the web application's authentication mechanism

Affected Products:

CWE:

566

References:

"Web Application Disassembly with ODBC Error Messages" (By David Litchfield)
SQL Injection Training Module

Technical Description:

The application uses a protection mechanism that relies on the existence or values of an input, but the input can be modified by an untrusted user in a way that bypasses the protection mechanism.

When security decisions such as authentication and authorization are made based on the values of user input, attackers can bypass the security of the software.

Suppose the query in question is:

```
SELECT COUNT(*) FROM accounts WHERE username='$user' AND password='$pass'
```

Where $user and $pass are user input (collected from the HTTP request which invoked the script that constructs the query - either from a GET request query parameters, or from a POST request body parameters). A regular usage of this query would be with values $user=john, $password=secret123. The query formed would be:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='secret123'
```

The expected query result is 0 if no such user+password pair exists in the database, and >0 if such pair exists (i.e. there is a user named 'john' in the database, whose password is 'secret123'). This would serve as a basic authentication mechanism for the application. But an attacker can bypass this mechanism by submitting the following values: $user=john, $password=' OR '1'='1.

Technical Description:

The application uses a protection mechanism that relies on the existence or values of an input, but the input can be modified by an untrusted user in a way that bypasses the protection mechanism.

When security decisions such as authentication and authorization are made based on the values of user input, attackers can bypass the security of the software.

Suppose the query in question is:

```
SELECT COUNT(*) FROM accounts WHERE username='$user' AND password='$pass'
```

Where $user and $pass are user input (collected from the HTTP request which invoked the script that constructs the query - either from a GET request query parameters, or from a POST request body parameters). A regular usage of this query would be with values $user=john, $password=secret123. The query formed would be:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='secret123'
```

The expected query result is 0 if no such user+password pair exists in the database, and >0 if such pair exists (i.e. there is a user named 'john' in the database, whose password is 'secret123'). This would serve as a basic authentication mechanism for the application. But an attacker can bypass this mechanism by submitting the following values: $user=john, $password=' OR '1'='1.

The resulting query is:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='' OR '1'='1'
```

This means that the query (in the SQL database) will return TRUE for the user 'john', since the expression 1=1 is always true. Therefore, the query will return a positive number, and thus the user (attacker) will be considered valid without having to know the password.

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

+ Threat modeling, + Attack surface analysis,…

Develop ("*make*") / Implement ("*buy*")

- *Source code control system, code reviews, daily builds, automated CASE tools…*

+ Developer security training, + Static analysis, + Secure code repositories,…

Testing/Validation

- *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*

+ Dynamic analysis, + Fuzzing,…

Release/Maintenance

- *Release testing*

+ Separation of duties, +Change management, +Operational practices…

Test Taking Tip

Focus on addressing each question individually

- As you take the test, if you don't know an answer, don't obsess over it
- Answer the best way you can or skip over the question and come back to it after you've answered other questions

Quiz

Agenda

- ✓ Introduction
- ✓ Software development life cycle (SDLC)
- ✓ SDLC and security
- ✓ Test taking tip
- ✓ Quiz