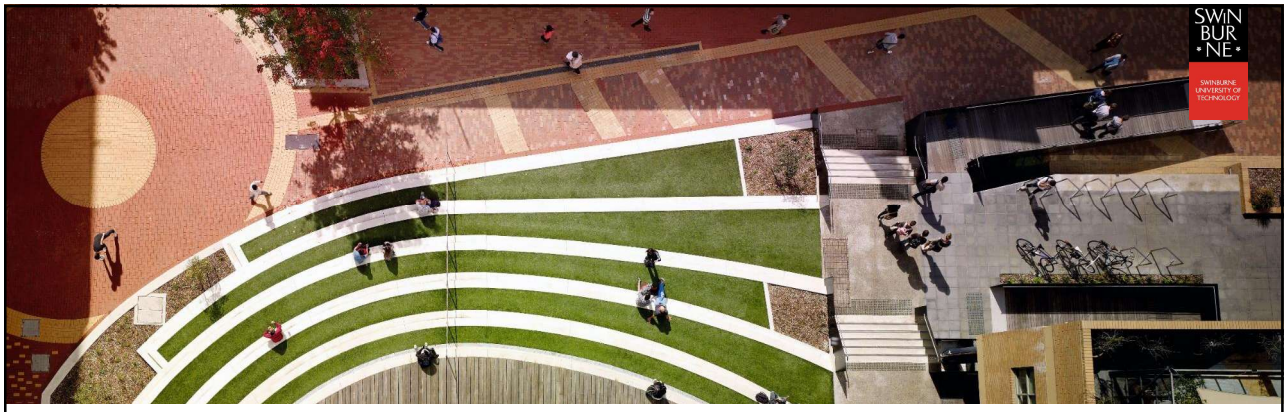


COS80022 – Software Quality and Testing

Week 11 – Non-Functional Testing

Outline

- Basic concepts
- Performance efficiency testing
- Reliability testing
- Security testing

What is Non-Functional Testing?

Non-Functional Testing

- Additional to the functional testing …
  - Functional testing checks *what* the system does
  - Non-functional testing identifies *how* well it does it (or at least should do it).
- Also can include how the system should be built
- Quality attributes of the system
- Mostly tested using black-box techniques as they are derived from specification requirements often defined as "constraints" of a system

Non-Functional Testing – Examples

- The maximum number of simultaneous users on a system are allowed
- System response times are acceptable
- Batch runs complete within an expected elapsed time
- Ensuring the integrity of a system from accident or malicious damage

Non-Functional Testing

- Requires specialised testing tools
- Technical knowledge
- Fully understanding the system complexities and supporting environment

- Not all types of non-functional testing are appropriate to every application
- Significant costs to do NFT correctly so need to consider risk

Non-Functional Testing

- Can occur at any test level
  - Performance benchmarking at unit test
  - Reliability testing at OAT and system test
- Order testing by risk priority and resource availability

NFT Considerations

- Requires considerable planning including risk assessment
- Remembering one of the 7 principles of testing

  "Testing is context dependent"
- Not every NFT quality characteristic and sub-characteristic is applicable every time there is a software or hardware change

Considerations

- Lifecycle timing
- Tools
- Software and documentation availability
- Technical expertise
- Resources
- Have a strategy for planning, preparation and execution of these unique test types

Measurements

- Metrics are usually tracked to form basis for SLAs
- Tests also may be executed after production release to verify the production environment
  - Efficiency
  - Reliability

Planning

- Stakeholder requirements
- Tools and training
- Test environment
- Organisational considerations
- Data security

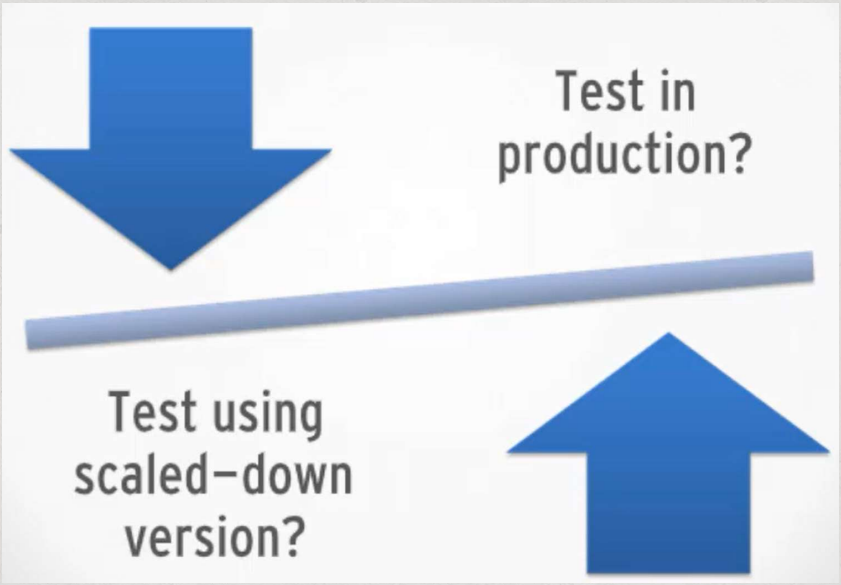
Stakeholder Requirements

- Determine the real non-functional requirements
  - Will using the current system as a benchmark be acceptable?
  - Do people know what they really need?
- Talk to the stakeholder and get multiple viewpoints
- Elicit, don't expect to be told

Tools and Training

- Tools for such items as security and performance testing must be acquired and learned
- Training for tools can be expensive
- Developing simulators is a development project
  - Simulators have to be tested too
  - Upgrades/retesting may be required
  - Safety-critical simulator use may require simulators to be certified

Test Environment



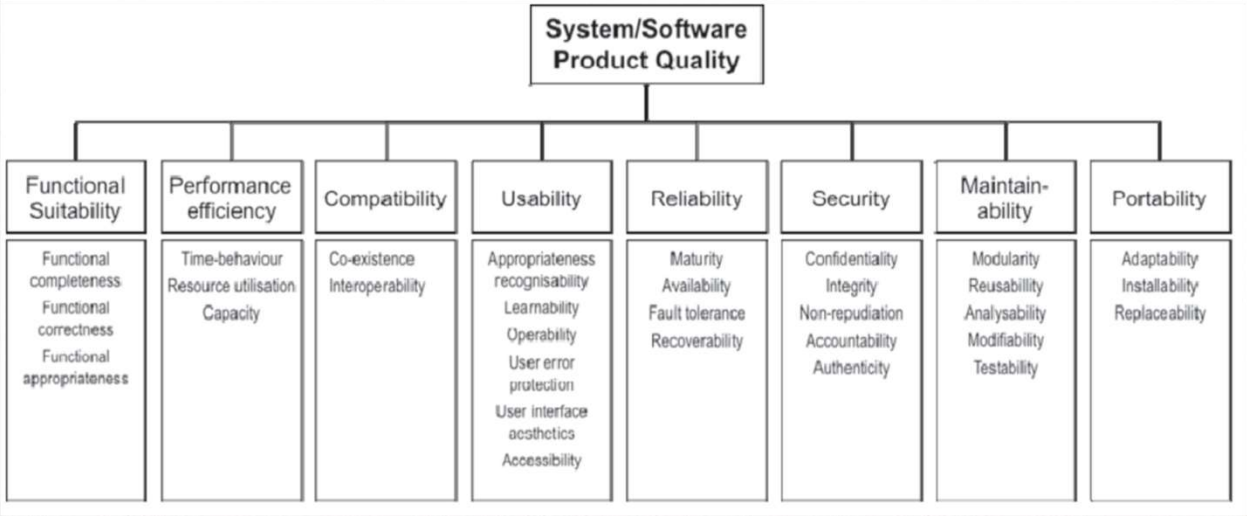
Organisational Considerations

- Who owns what
- Where is it located
- When is it available
- Does teaching need to be coordinated across groups/ organisations
- Is availability an issue with borrowed resources or experts “on call”?

Data Security

- Data encryption may be required – complications for verifying output
- Data anonymisation may be needed
- May not be able to generate test data from production data (data protection laws, privacy considerations)

Types of NFT



Performance Efficiency Testing

Performance Efficiency Testing

- To check how fast a specific aspect of a system performs under a particular workload
- Can serve to demonstrate that the system meets performance criteria
- Can compare two systems to find which performs better
- Can measure what part of the system or workload causes the system to perform badly

Performance Efficiency Testing

Performance testing cares about:

- Time behavior
  - Response time, tasks performed / unit time

Resource testing looks at:

- Utilisation of resources
  - CPU utilization, memory usage (RAM, ROM, cache, disk space)
  - Typically percentages of max. available

Capacity testing:

- Checking the degree to which the maximum limits of a product or system parameter meet requirements

Performance Efficiency Testing

Load testing :

- To determine a system's behavior under both 'normal' and anticipated peak load conditions (as defined in requirements documentation)
  - A response time no greater than 2 seconds to perform transaction X with max. # of concurrent users, 120 other users performing a 'typical' set of actions (operational profile) on a default configuration/environment

Performance Efficiency Testing

Stress testing :

- Not concerned with whether a specific requirement (level of resource utilization or response time) is achieved or not
- But rather a measurement activity
- Interested in how the system reacts to (normally) ever-increasing loads (beyond those required) until it:
  - Exhausts system resources
  - Fails to respond at all (crash/hang)

Performance Efficiency Testing

Scalability testing:

- Testing a system ability to grow and the assessment of whether to 'scale up' or 'scale out'

Soak testing:

- Run over an extended period of time mostly applying 'normal' load e.g. one week, to see how the system performs
- Often looks for memory leaks that are not evident under shorter functional performance tests

Performance Efficiency Testing

Spike (& Bounce) testing:

- Generating a sudden increase in the load to a very large number of users or transactions or both to observed the system behavior
- Used to determine whether performance will degrade or the system will fail
- Bounce testing is repeating the process to that load is increased and decreased and increased again

Performance Efficiency Testing Considerations

- Need to obtain answers to a range of questions to firstly determine the operational profile of the system
- Ensure stakeholders are aware of the slight differences in the various performance efficiency types when communicating what is to be tested

When should Performance Efficiency Testing be Done?

- Typically requires complete system – so final run usually included as part of system testing
- Although some may be performed during component testing
  - If possible do this for critical components
  - Requires use of stubs and drivers

What's Needed for Performance Efficiency Testing?

- Needs depend primarily on the load to be generated, which may be based on the number of virtual users to be simulated and the amount of network traffic they are likely to generate
- Enough virtual users
  - Needs specialist tools compatible with system network protocols and infrastructure
  - Needs to be budgeted for
  - May need to rent required infrastructure (e.g. cloud)

Performance Testing

- Measures response time to user inputs
- Measures response time to other system inputs (transaction requests, file uploads, etc.)
- Must consider the specific conditions for the system when the measurement is taken
- Operational profiles are important for defining the system conditions and environment
- Performance measurements are interesting as a testing objective during load, stress and scalability testing

Load Testing

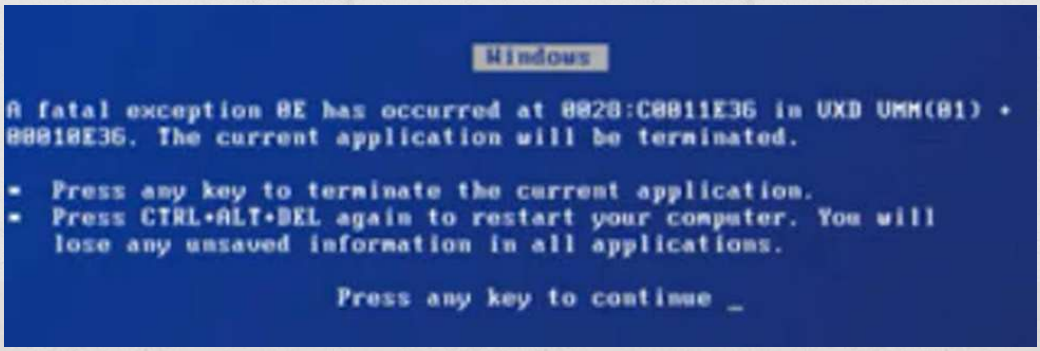
- Focus is on realistic load
- Load is usually stepped up to watch the software handle the increases
- Sources include users and other interfacing systems
- Background processing of batch files and transactions contribute to system load

Load Testing – Objectives

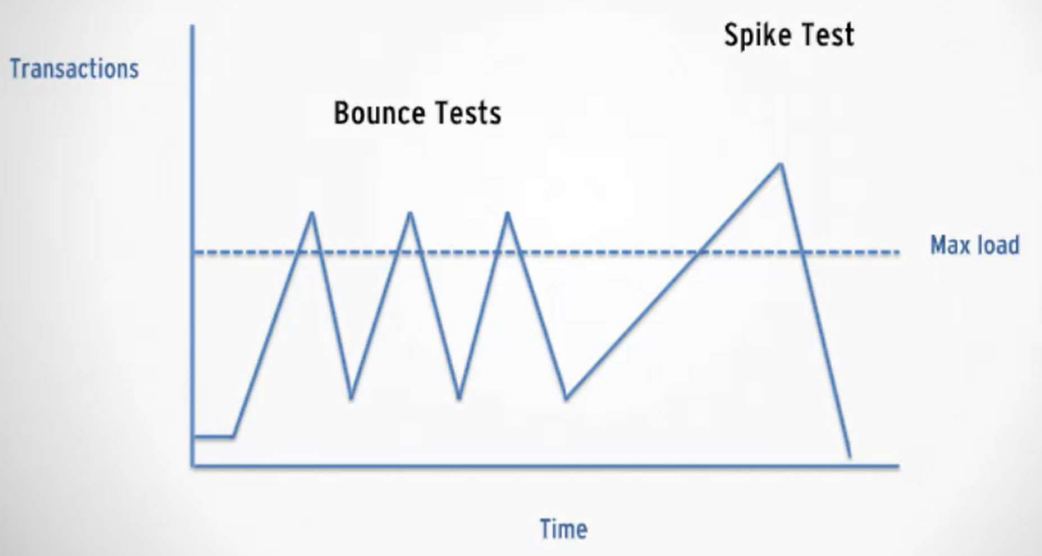
- Ability to handle multiple users' concurrent transactions
- Session integrity (none lost of crossed up)
- Performance (response to a user, processing of a transaction)
- Resource behaviour (buffers, queues, databases, connections, volume-related issues, memory, network bandwidth)

Stress Testing

- Testing about the specified limits
- Determine the “failure point”
- Looking for graceful degradation



Spike and Bounce



Scalability Testing

Goal

- Will the system be able to handle planned growth
- Can the system handle success

Dealing with production

- Often conducted in production
- Requirements are often not known until the product has been in production for some time

Resource Utilisation Testing

Concurrent testing

- Resource utilisation evaluation if often conducted during other load, stress and scalability tests
- Similar to performance testing – it's an evaluation while the system is running

Resources

- Include memory, disk space, table space, connection management, network bandwidth
- Dynamic testing is required

Performance Efficiency Testing

- Measurement and monitoring
- Planning
- Requirements
- Test approach
- Tools
- Environments
- Organisational considerations
- Lifecycle considerations
- Specification and execution

Measurement and Monitoring

Measurements

- What to measure
- Precision requirements
- Cost vs. benefit of taking the measurements
- Data for follow up activities

Monitoring

- Duration of monitoring
- Data gathering for data analysis

Planning

Identify and assess risk

- Will the design handle the expected load?
- Is the network enough?
- Are the servers configured optimally

Identify the test objects

- Concentrate on a subset of the system (particularly components or architectures)
- Consider the complete system
- Focus on time-critical components

Requirements

Rarely stated

- May have to review architecture and design to find the “hidden” requirements
- Extract, clarify and specify the requirements in a testable manner

Need to know

- Measurable response time
- Operational profile at time of execution
- What % of the time the response time has to be achieved
- System configuration, data to be used

Test Approach

- Base tests on operational profiles
- Don’t forget the static testing
- Consider tools and benchmarks
- Focus where possible, broaden scope as needed

39

Tool Considerations

- Simulators / virtual users
- Cost
- Buy or build
- Skills and training requirements
- Reporting capabilities

40

Environment Considerations

- Load
  - The test environment has to support the load you will generate
  - Capacity will be needed for volume tests
- Production-like
  - The system needs to be representative of production
  - Don't test on dedicated servers and deploy on VMs
- Testing for the Internet
  - Be sure the test network has noise
  - Background load generation on the network may be needed

Organisational Considerations

System and component ownership

- System administration of test systems
- Availability at the time it's needed

Utilising third party resources

- Use a test lab's environment
- Cloud
- Repeatability and availability

Lifecycle Considerations

- Schedule tests early in cycle
  - Early feedback can affect design
  - Time to correct faults
  - Time critical items can be tested early (unit test)
- Schedule tests later in cycle
  - Less repetition needed, less cost
  - All components available, drivers and stubs not needed
  - Less time spent working around defects
- Consider concurrent testing

Specification

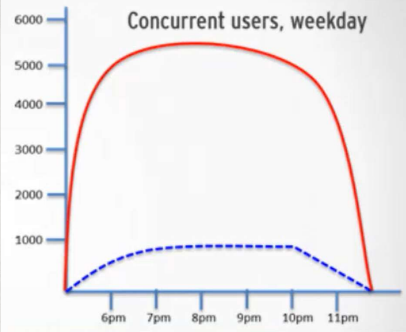
- Operational profiles
  - Each profile represents how a particular user will interact with the application
  - Usually there are multiple uses across different types of users
  - Defining the profiles requires considerable analysis
- Script creation
  - Operational profiles are used to detail the activities and the frequency of those activities
  - Activities form the basis of the test scripts
  - Frequency forms the basis of the overall test (how many, doing what, when)
- Human users may not be the only type of “user”

Example – Operational Profiles

- Given the information in the next slide, create a usage diagram for the profile viewing transactions on an average weekend. From that diagram, determine how many viewing transactions should be planned during the test and when those transactions should occur.

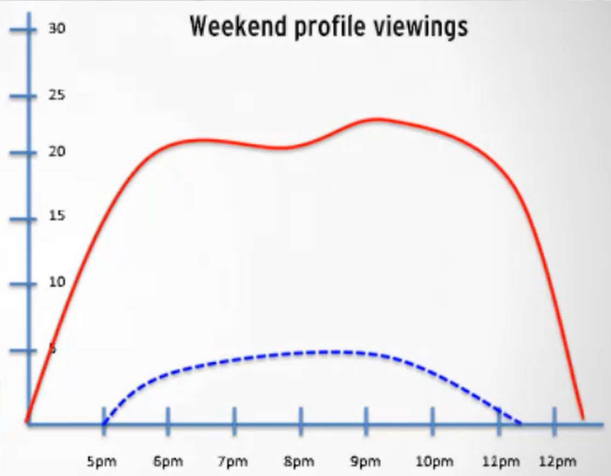
Operational Profiles – Date4You

- Waiters:
  - online an average of 4 hours per evening, all days
  - Average 20 times per session, investigate a profile that has queried
  - Normal evening has 1000 concurrent waiters
- Seekers:
  - Online an average of 5 hours per evening, 8 hours on weekend evenings
  - Query an average of 20 profile viewings per hour
  - Normal evening has 5000 concurrent seekers and 8000 seekers on weekends



Hour	Waiters	Seekers
1	500	4800
2	800	5200
3	1000	5300
4	1000	5100
5	1000	5000
6	300	3000

Weekend Profile Viewings



Hour	Waiters	Seekers
1 (5pm)	0	15
2	4	20
3	5	21
4	5	19
5	6	23
6	4	22
7	1	19
8	0	8

Execution

- All the setup it done, but
  - Need to monitor during execution to see if the load is actually being generated
  - Tuning or script changes may be needed
- During execution
  - Monitor carefully
  - Track system measurements against load
- Watch for errors
  - Incomplete transactions
  - Data issues
  - Correct and retest

Reliability Testing

Reliability Testing

- Checking the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time
  - Maturity (Robustness), recoverability, availability, fault tolerance
- Reliability is based on the probability that a system will break i.e. the more likely it is to break, the less reliable it is

Reliability Testing

Maturity Testing

- Need to understand environmental factors that could lead to failures, e.g.
  - Lack of memory
  - Missing libraries (DLLs)
  - Missing files
- Several factors to assessing the maturity of a system
  - Probability of Failure on Demand (POFOD)
  - Mean Time to Failure or Mean Time between Failures (MTTF/MTBF)
- Tools can be used to create scenarios of system resources being withdrawn or restricted

Reliability Testing

Recoverability testing

- Failover testing
- Backup & restore testing
- Disaster recovery testing

Availability testing

- Probability the system will be functioning correctly at any given time

Reliability Testing

Fault tolerance testing

- Decrease in quality proportional to the severity of the failure
- The ability of a system to “degrade gracefully”
- Safety-critical and business-critical systems are typically fault tolerant
- Testing is concerned with how the remainder of the system can cope if a component fails
- Tested by fault injection or simply “unplugging” of hardware components

Reliability Testing Considerations

- Normally part of operational acceptance testing (OAT)
- Reviews of documentation for back-up and restore activities
- Checking the back-u[s are actually occurring according to plan
- Dry runs through the critical restore procedures
- Dynamic tests of full or partial restores
  - To check speed of restores
  - To check ‘freshness’ of data restored

Maturity

Two factors in maturity

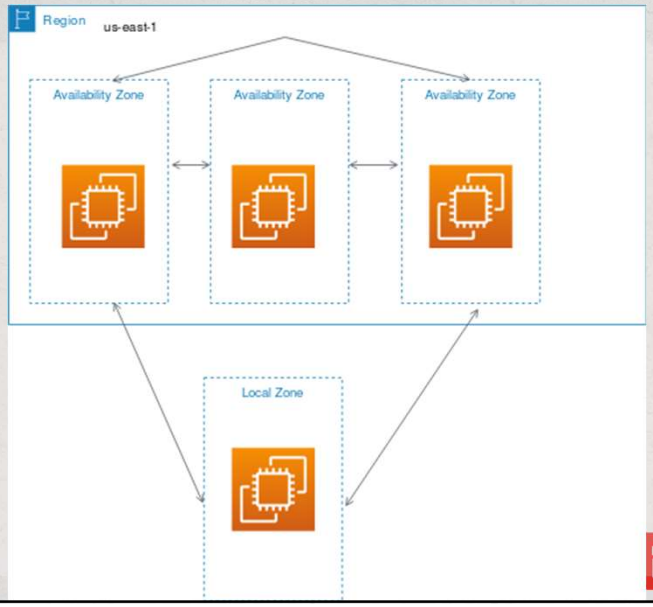
- What is the system doing (condition)
- How long has it been doing it (time or operation)

Measurements are based on failure intensity

- MTBF (mean time between failures)
- MTTR (mean time to repair)
- Number of high severity failures in a time frame

Availability

- Example from AWS
  - Region
  - Availability Zone
  - Local Zone
  - EC2: four 9s
  - S3: eleven 9s



Fault Tolerance

- What happens
  - When a failure occurs
  - When an unexpected event occurs (e.g. disk full)
- Maintaining a specific level of performance
- Robustness and error tolerance

Recoverability

- Recoverability measurements
  - Recovering from software/hardware failure
  - Re-establish level of performance (meet SLAs)
  - Recover any data affected
- Failover
  - Failure is induced or simulated
  - Failover is needed when functionality cannot be interrupted
  - Graceful failover should be invisible to the user
  - May required additional hardware (load balancers, redundant systems) to supply failover capabilities
  - Redundant dissimilar systems can be duplex, triplex or quadruplex
  - Voting software determines which software to trust in a fail situation

Recoverability

- Backup and restore
  - Focuses on procedural measures to take backups
  - Failures to necessitate restoring from backups
  - Technical reviews are often used for the procedures
  - OATs are used to validate the procedures
- Measures
  - Time to create the backup (e.g. incremental, full)
  - Time to restore from the backup
  - Level of backup achieved (e.g. no more than 4 hours lost)

Security Testing

Security Testing

- Checking the degree to which a product or system protects information and data to that persons or systems have data access appropriate to their types and levels of authorization
  - Confidentiality; Integrity; Non-repudiation; Accountability; Authenticity
- To find out all the potential loopholes and weaknesses which might result in loss/theft of highly sensitive information or destruction of the system by an intruder

Security Testing

- Security describes the attributes of the software that affect its ability to prevent unauthorised access, whether accidental or deliberate, to programs and data
- Security testing differs from functional testing in two ways:
  - The data used for security testing is designed to identify security flaws
  - There is a wide variety of security defects and these are not always clearly identifiable. Effective security testing requires a special skill set.

Types of Security Testing

Discovery:

- Identify systems within scope and the services in use

Vulnerability scan:

- Look for known security issues by using automated tools to match conditions with known vulnerabilities

Vulnerability assessment:

- Use discovery and vulnerability scanning to identify security vulnerabilities and places the findings into the context of the environment under test

Types of Security Testing

Security assessment:

- Build upon vulnerability assessment by adding manual verification to confirm exposure
- But not include the exploitation of vulnerabilities to gain further access

Penetration testing:

- Simulate an attack by a malicious party
- Build on the previous stages and involve exploitation of found vulnerabilities to gain further access

Types of Security Testing

Security audit:

- Driven by an Audit / Risk function to look at a specific control or compliance issue

Security review:

- Verification that industry or internal security standards have been applied to system components or product

65

Security Testing

- Most specialised and technical in non-functional testing area
- Scheduled during component, integration and system testing levels and preferably on a regular basis once live
- Defects found are specific to the security of a system
- Improvements made to the security of a system may affect its performance
  - Advisable to repeat performance tests if any changes made due to correcting security defects

66

Security Testing

- Threats models should be created as early as possible in the life cycle and should be revisited as the application evolves and development progresses
- Static analysis tools contain an extensive set of rules which are specific to security threats and against which the code is checked

Security – Discussion

- 3A in identify access security
  - Authentication: Confirm the right person is accessing the system
  - Authorisation: Give minimal access for their needs
  - Accounting: Monitor activities for security and compliance

Security – Discussion

- Threats
  - Unauthorised access
  - Unexpected side-effects
  - Cross-site scripting (XSS)
  - Buffer overflow
  - Denial of Service (DOS)
  - Man-in-the-middle
  - Logic bombs

Unauthorised Access

- Access via special programs or viruses
  - Programs are specially developed by hackers to gain access
  - Reside on the local machine and send information back to hacker
  - May also execute code included in the virus to damage the system
- Passwords obtained by illicit means
- Violation of user rights
  - Certain users have certain permissions
  - If the wrong person can see the wrong data → security problem

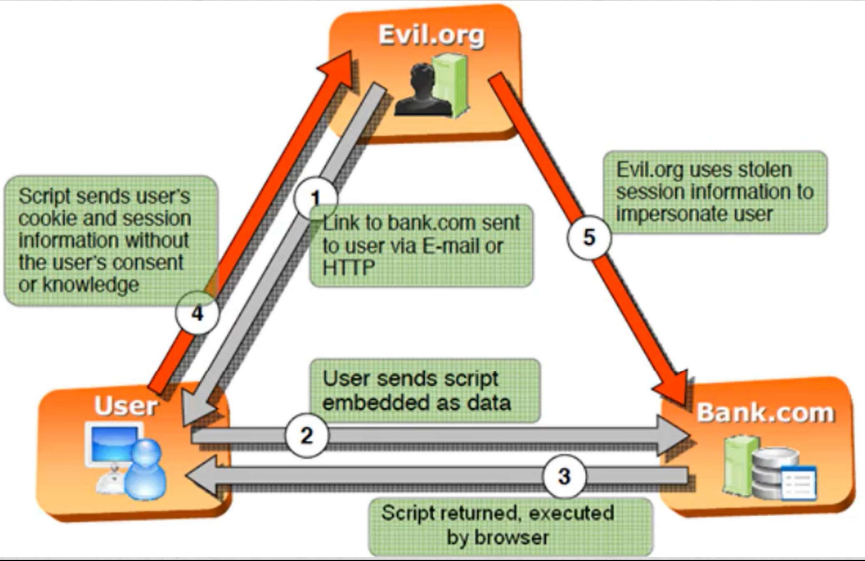
Unexpected Side Effects

- Cookies
  - Cookies or similar session related data might be preserved across sessions and across users
  - Particularly a concern with public machines
- Unprotected data
  - When data is transferred or stored, it may be accessible
  - Phones may take screen snapshots to track multiple sessions
- Free-text passing of passwords or URLs

Cross-Site Scripting (XSS)

- Web pages
  - If the vulnerability exists, code can be inserted into a web page
  - This code can be executed by subsequent "knowledgeable" users
  - If malicious, damage can be done to the back end servers and databases
- Easily detected with security scanning tools

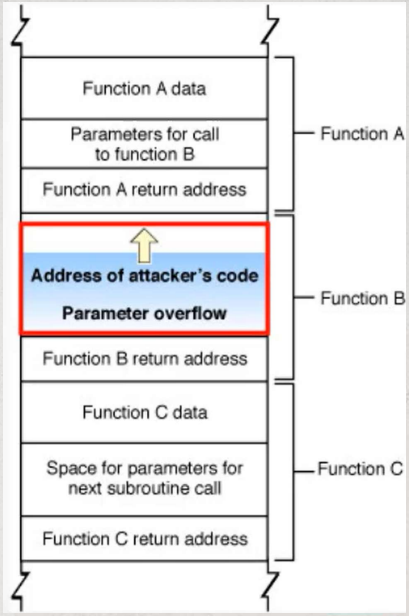
XSS



From: www.hackerschronicle.com

Buffer Overflow

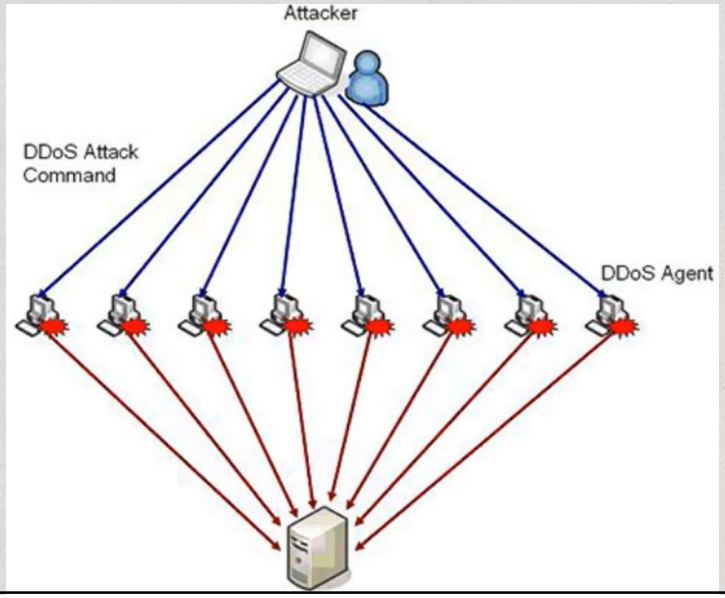
- Still one of the favourites!
  - Any software that accepts inputs is vulnerable
  - All input fields must be size contained
- Buffer exceeded
  - When the expected buffer is exceeded (and not detected) the "overflow" writes over the next information in memory
  - This overflow may contain executable code, corruption or cleverly crafted misinformation



Man-in-the-Middle

- Code in the middle
  - A message sent from the user to the server is intercepted
  - The message is modified and then sent on its way
- Works both ways
  - Messages can be intercepted going either way
  - Changes are made to favour the hacker in some way

Denial of Service



Logic Bombs

- Code that executes if:
  - Sometimes happens on a particular date
  - Sometimes executes if something else doesn't happen
- Job security?
  - Sometimes planted in the code to execute if the developer doesn't log in or reset something
  - Can have devastating results

Other Types of Non-Functional Testing

Functional Suitability Testing

Functional completeness:

- Testing that all the specified tasks and user objectives have been fulfilled

Functional correctness:

- Testing that the system adheres to specified requirements including computations to the required level of precision

Functional appropriateness:

- Evaluating and validating the suitability of a set of functions to accomplish its intended specified tasks and objectives

Functional Suitability Testing

Quality Attribute	Techniques to Test	When tested in SDLC	Typical Defects found
Functional Completeness	Most black-box techniques Maintaining a coverage traceability matrix	Throughout the SDLC	Missing required functionality in delivered system
Functional Correctness	Most black-box techniques	Throughout the SDLC	Accuracy of delivered requirements not met Incorrect handling of data or processes
Functional appropriateness	Use case testing Checklists	System Testing UAT	Not "Fit for purpose" System is not meeting the needs of the user in a way that is considered as acceptable

Compatibility Testing

- Checking the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment
  - Co-existence
  - Interoperability

Compatibility Testing

- Co-existence testing
  - Order of installation
  - Concurrent use
  - Order of instantiation
- Interoperability testing
  - Need to consider all the intended target environments
  - Exercise the various data exchange points
  - Can focus on use of communications standards such as XML and automatically detect the communications needs to the systems it interacts with and adjust accordingly
  - Important when dealing with 3rd party COTS software

Compatibility Testing Considerations

- May need referring to system architecture and design documents to fully understand interaction points
- Decision tables, state transition diagrams and use cases can be used

Usability Testing

- The ease by which all likely users can use or learn to use the system to reach a specified goal in a specified context
- Testing will measure the following:
  - Effectiveness
  - Efficiency
  - Satisfaction
  - Attractiveness

Usability Testing

- Appropriate Recognisability
- Learnability
- Operability (example next slide)
- User Error Protection
- User Interface Aesthetics
- Accessibility

Operability Testing

- Should be done under conditions as close as possible to live production
- Use of an operability lab, set of scripts for “users” to perform
- Allow users to experiment with the software to be observed
- Operability tests also performed by the test analyst during functional system testing

Operability Testing

- A set of guidelines to operability testing helps ensure consistency in identifying defects
  - Accessibility of instructions
  - Clarity of prompts
  - Number of clicks to complete an activity
  - Error messaging
  - Processing indicators
  - Screen layout guidelines
  - Use of colours and sounds
  - Any other factors that affect the user's experience

Usability Testing

Test considerations for usability – principal techniques are:

- Inspection, evaluation or review
- Dynamically interacting with prototype
- Performing verification and validation of the actual implementation
- Performing surveys and questionnaires

Maintainability Testing

- Degree of effectiveness and efficiency with which the product or system can be modified by the intended maintainers
  - Modularity; reusability; analysability; modifiability; testability
- Requires good knowledge of:
  - The software
  - The architectural design
  - The hardware

Maintainability Testing

- Goals
  - Minimise cost of ownership or operation
  - Minimise down time required for maintenance
- Need to plan for maintainability tests
  - Software changes are likely after the software goes into production
  - Benefits of achieving the goals outweigh the costs of performing the tests and making changes
  - Risks of poor maintainability justify the testing

Maintainability Testing

- Must plan early
  - Much of the testing involves design reviews and static analysis
  - Maintainability is built into the code and documentation
  - Dynamic testing includes testing the documented procedures developed to maintain the software (may be a part of OAT)
- Gathering information
  - Structural complexity; nesting levels of the code; comments and documentation; lines of code, numbers of methods
- Compare information to standards

Portability Testing

- Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another
  - Adaptability; Installability; Replaceability
- Objectives of Portability Testing
  - Partially validate the system
  - Determine if the system can be ported to each of its required environments
  - Determine if look and feel of webpages are similar and functional on various browser types and versions

Portability Testing – Planning

- Goals
  - Ensure the software will work in the target and anticipated environments
  - Verify that the documented procedures for porting the software are accurate
- Planning for portability tests
  - Risks must be assessed to determine likely environments and long term need for portability
  - Potential configurations pool is enormous – need to be realistic
  - Important to understand the anticipated longevity of the software
  - Portability tests can often be paired with other tests, including integration and installability
  - Special skills may be required to configure alternate environments or components
  - API tests for the interfaces may be critically important

Portability Testing – Execution

- Static testing
  - Architecture reviews
  - Design reviews
  - Interface code reviews
- Dynamic testing
  - Multiple environments or simulators required
  - Need to be able to configure environments and code to test the new components
  - Need realistic running environment to determine proper compatibility
  - Some tests, such as installability, will come late in the schedule

Portability Testing

- Reviews of compliance to specifications
- Component integration testing to check functionality is maintained
- Other quality attribute testing, such as performance efficiency

Summary

- Considerations of non-functional testing
- Different types of non-functional testing
  - Performance efficiency testing
  - Security testing
  - Reliability testing

References

- ISO/IEC 25010:2011. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*

Next

- Tutorial
  - Lab Work: Practice JMeter
- Next week
  - Agile testing