YAZILIM TESTİ VE PROJE YÖNETİMİ

Yazılım Testlerinin Sınıflandırmaları

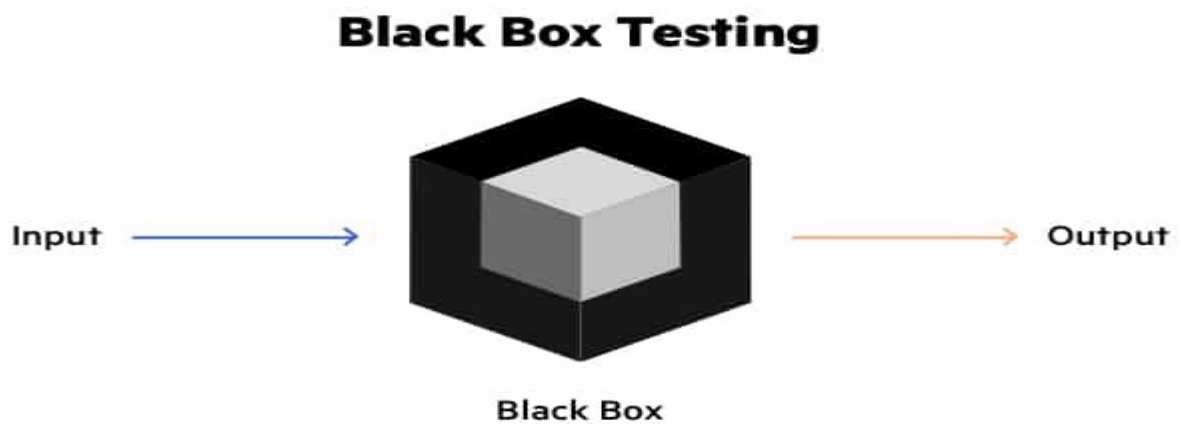
2024 BAHAR

29.04.2024

Test Tekniklerinin Sınıflandırması

- ❑Black-Box Testing
- ❑White-Box Testing
- ❑Gray-Box Testing

- ❑Manual versus Automated Testing
- ❑Static versus Dynamic Testing



- Kara kutu testi, bir sistemin iç yapısı ile ilgili bilgi gerektirmeyen test yapılmasıdır.
- Test aygıtı ile bir girdi sağlanır ve test edilen sistem tarafından üretilen çıktıyı gözlemlenir.
- Kara kutu testi ile sistemin beklenen ve beklenmeyen durumlara nasıl tepki verdiğini, sistemin yanıt süresini, kullanılabilirlik sorunlarını ve güvenilirlik sorunlarını tanımlamayı mümkün kılar.
- Test uzmanlarının sistemin uygulama ayrıntılarını öğrenmesine gerek yoktur
- Test uzmanlarının teknik bilgi, tasarım ve programlama becerilerine ihtiyacı yoktur
- Kara kutu testi ile Yanlış Pozitif sonuç elde edilmesi düşük olasılıklıdır
  - Yanlış Pozitif, veri değerlendirmede bir ölçümdür.
  - Test sonucunun belirli bir durumun (örneğin bir hastalığın) varlığını, gerçekte olmadığı hâlde yanlış olarak onaylaması (pozitif göstermesi) şeklinde ortaya çıkan hatadır

Kara Kutu ve Beyaz Kutu Testi

- Birçok uygulamada kara kutu testini beyaz kutu testiyle birleştirilir.
- Beyaz kutu testi, bir uygulamanın kaynak kodu, mimarisi ve yapılandırmasına ilişkin ayrıntılı iç bilgilerle test edilmesidir.
- Kara kutu testi kapsamlı olarak gerçekleştirilemeyebilir veya
- Kara kutu testinin test edemediği güvenlik açıkları, veri akışı sorunları ya da yolları takip sorunları olabilir.
- Kara kutu testi yetersiz kaldığında kara kutu ve beyaz kutu testleri birleştirilir.
  - Böylece yazılım uygulamasının kapsamlı olarak "içten dışa" incelenmesi mümkün olabilir.
    - ✓ Geliştirilen ürünü kalite ve güvenlik alanlarında güçlendirir.

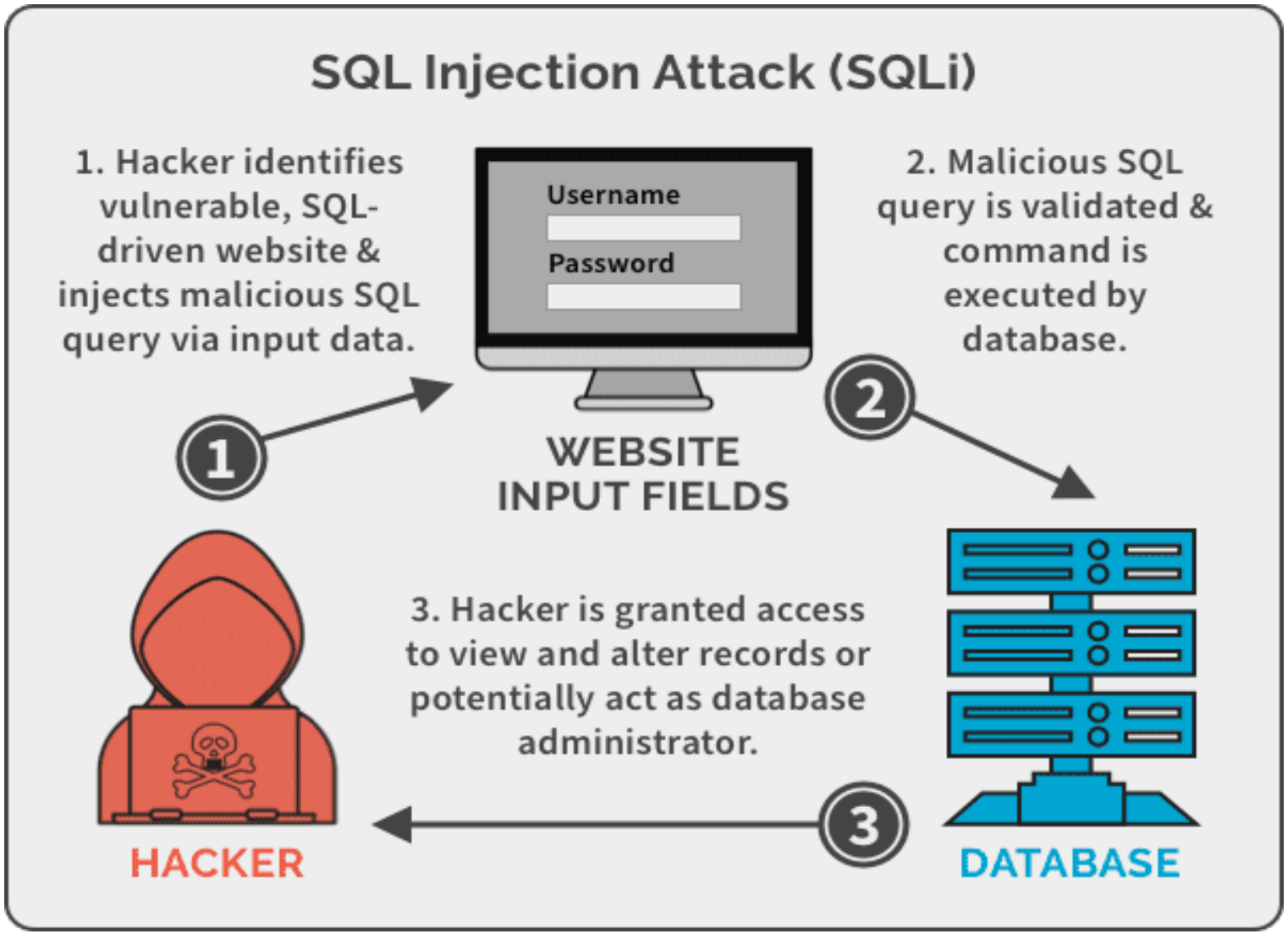
Gri Kutu Testi

- Beyaz kutu testinde, test uzmanı kodun iç yapısı ile ilgili olarak tam bilgi sahibidir.
- Kara kutu testi ise, kod bilgisi olmadan kullanıcının bakış açısına göre gerçekleştirilir.
- Gri kutu testinde ise testinin çalışan koda erişimi sınırlıdır.
  - Uygulamalar ve ortamlar, kodun iç yapısı ile ilgili çalışmalarda kısmi bilgiyle test edilir.
  - Gri kutu testi yaygın olarak penetrasyon (sızma) testi, uçtan uca sistem testi ve entegrasyon testi için kullanılır.

- Interactive Application Security Tools (IAST) araçları ile gri kutu testi yapılabilir.
- IAST araçları, kodu değerlendirmek amacıyla beyaz kutu testinde kullanılan DAST (Dynamic Application Security Testing) ile Static Application Security Testing (SAST) yapısını birleştirir.
- IAST araçları, test uzmanlarının ve geliştiricilerin çalışmalarını birleştirmesine ve testin verimli şekilde sonuçlanmasını sağlar.

IAST Araçları (tools)

- Bazı IAST çözümleri, uygulama oluşturulduktan sonra kullanılmakta olan kodu tarar.
- Bazı IAST çözümleri, DevOps ekiplerinin "güvenliklerini sola kaydırmalarına /shift the security left" imkan verir
- Bazı IAST çözümleri güvenlik açıklarının ve hataların daha kolay/ucuz düzeltilmesi için geliştirme aşamasında kodların test edilmesini sağlayan IDE /Integrated Development Environment ile entegrasyon sağlar.
- Sonuç olarak DevOps ekipleri, IAST'yi herhangi bir aşamada geliştirme yaşam döngüsüne entegre ederek, uygulama dağıtılmadan önce tüm güvenlik açıklarını keşfedip düzeltebilir.
  - SQL injection, API anahtarlarının açık metin olarak sabit kodlanması veya şifrelenmemiş bağlantılar gibi….


SQL Injection Attack (SQLi)
1. Hacker identifies vulnerable, SQL-driven website & injects malicious SQL query via input data.
Username
Password
WEBSITE INPUT FIELDS
2. Malicious SQL query is validated & command is executed by database.
1
2
3
3. Hacker is granted access to view and alter records or potentially act as database administrator.
HACKER
DATABASE

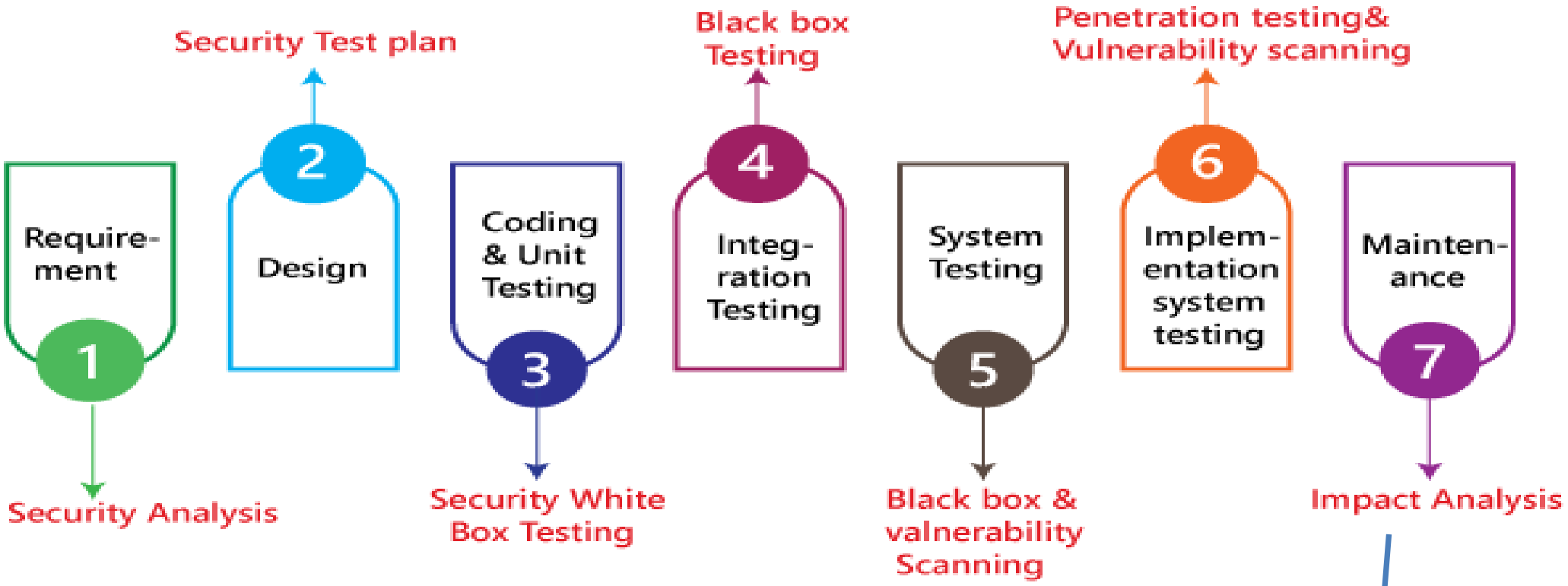
	Black-Box *aka close box penetration testing*	Grey-Box *combination of black box and white box testing*	White-Box *aka open box penetration testing*
Goal (Amaç)	Mimic a true cyber attack	Assess an organization's vulnerability to insider threats	Simulate an attack where an attacker gains access to a privileged account
Access Level (Erişim Düzeyi)	Zero access or internal information	Some internal access and internal information	Complete open access to applications and systems
Pros (Artıları)	Most realistic Testing is performed from point of view of attacker	More efficient than black-box andsaves on time and money Testing is performed from point of view of attacker	More comprehensive, less likely to miss a vulnerability and faster Testing is performed from point of view of attacker
Cons (Eksileri)	Time consuming and more likely to miss a vulnerability	No real cons for this type of testing	More data (ex, source code) is required to be released to the tester and more expensive

- Regresyon testi, önceden geliştirilen ve test edilen yazılımın bir değişiklik sonrasında hala beklendiği gibi performans gösterdiğinden emin olmak için fonksiyonel ve nonfunctional testlerin yeniden çalıştırılmasıdır.
- Sistemin herhangi bir yerindeki değişiklik diğer kısımlarını (kod parçalarını) etkiliyorsa yeniden test gerekir. Bu «regresyon testi »dir.

	Black Box	White Box
Functional	Integration Testing System Testing Acceptance Testing Regression Testing	Code Tests (Unit-Testing) Regression Testing
Nonfunctional	Performance Testing Accessibility Testing (Penetration Testing) (Security Testing)	Penetration Testing Security Testing

- Sızma (penetration) testi, geliştirilen ürünün güvenliğinin kontrolü için bilgisayar sistemine saldırı simüle edilmesidir.
- Sistemdeki zayıflıkların etkilerini bulmak için saldırganlarla aynı araçlar, teknikler ve süreçler kullanılır.
- Sistemin, kimliğin doğrulanmış / doğrulanmamış olduğu durumlardan ya da sistemin çeşitli işlevlerinden/rollerinden gelen saldırılara dayanacak kadar sağlam olup olmadığı incelenir.

Security Testing along with SDLC



Bilgi sistemindeki değişikliklerin sistemin güvenlik durumunu ne ölçüde etkilediğini belirlemek için gerçekleştirilen analizdir.

Test Tekniklerinin Kategorizasyonu-I

Technique	*Manual*	*Automated*	*Static*	*Dynamic*	*Functional*	*Structural*
Acceptance testing	x	x		x	x	
Ad hoc testing	x				x	
Alpha testing	x			x	x	
Basis path testing		x		x		x
Beta testing	x			x	x	
Black-box testing		x		x	x	
Bottom-up testing		x		x		x
Boundary value testing		x		x	x	
Branch coverage testing		x		x		x
Branch/condition coverage		x		x		x
Cause–effect graphing		x		x	x	
Comparison testing	x	x		x	x	x
Compatibility testing	x	x				x
Condition coverage testing		x		x		x
CRUD (create, read, update, and delete) testing		x		x	x	
Database testing		x		x		x
Decision tables		x		x	x	
Desk checking	x			x		x
End-to-end testing	x	x			x	

Test Tekniklerinin Kategorizasyonu-II

	Manual	Automated	Static	Dynamic	Functional	Structural
Run charts	x		x		x	
Sandwich testing		x		x		x
Sanity testing	x	x		x	x	
Security testing	x	x				x
State transition testing		x		x	x	
Statement coverage testing		x		x		x
Statistical profile testing	x		x		x	
Stress testing	x	x		x		
Structured walkthroughs	x			x	x	x
Syntax testing		x	x	x	x	
System testing	x	x		x	x	
Table testing		x		x		x
Thread testing		x		x		x
Top-down testing		x		x	x	x
Unit testing	x	x	x			x
Usability testing	x	x		x	x	
User acceptance testing	x	x		x	x	
White-box testing		x		x		x

Test Teknikleri ve Kısa Açıklamaları - I

Technique	*Brief Description*
Acceptance testing	Final testing based on the end-user/customer specifications, or based on use by end users/ customers over a defined period of time
Ad hoc testing	Similar to exploratory testing, but often taken to mean that the testers have significant understanding of the software before testing it
Alpha testing	Testing of an application when development is nearing completion; minor design changes may still be made as a result of such testing. Typically done by end users or others, not by programmers or testers
Basis path testing	Identifying tests based on flow and paths of a program or system
Beta testing	Testing when development and testing are essentially completed and final bugs and problems need to be found before final release. Typically done by end users or others, not by programmers or testers
Black-box testing	Testing cases generated based on the system's functionality
Bottom-up testing	Integrating modules or programs starting from the bottom
Boundary value testing	Testing cases generated from boundary values of equivalence classes
Branch coverage testing	Verifying that each branch has true and false outcomes at least once
Branch/condition coverage testing	Verifying that each condition in a decision takes on all possible outcomes at least once
Cause–effect graphing	Mapping multiple simultaneous inputs that may affect others, to identify their conditions to test
Comparison testing	Comparing software weaknesses and strengths to competing products

Test Teknikleri ve Kısa Açıklamaları - II

Compatibility testing	Testing how well software performs in a particular hardware/software/operating system/network environment
Condition coverage testing	Verifying that each condition in a decision takes on all possible outcomes at least once
CRUD testing	Building a CRUD matrix and testing all object creations, reads, updates, and deletions
Database testing	Checking the integrity of database field values
Decision tables	Table showing the decision criteria and the respective actions
Desk checking	Developer reviews code for accuracy
End-to-end testing	Similar to system testing; the "macro" end of the test scale; involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate
Equivalence partitioning	Each input condition is partitioned into two or more groups. Test cases are generated from representative valid and invalid classes
Exception testing	Identifying error messages and exception-handling processes and conditions that trigger them
Exploratory testing	Often taken to mean a creative, informal software test that is not based on formal test plans or test cases; testers may be learning the software as they test it
Free-form testing	Ad hoc or brainstorming using intuition to define test cases
Gray-box testing	A combination of black-box and white-box testing to take advantage of both
Histograms	A graphical representation of measured values organized according to the frequency of occurrence; used to pinpoint hot spots

Test Teknikleri ve Kısa Açıklamaları - III

Incremental integration testing	Continuous testing of an application as new functionality is added; requires that various aspects of an application's functionality be independent enough to work separately before all parts of the program are completed, or that test drivers be developed as needed; done by programmers or by testers
Inspections	Formal peer review that uses checklists, entry criteria, and exit criteria
Integration testing	Testing of combined parts of an application to determine if they function together correctly. The "parts" can be code modules, individual applications, or client/server applications on a network. This type of testing is especially relevant to client/server and distributed systems
JADs	Technique that brings users and developers together to jointly design systems in facilitated sessions
Load testing	Testing an application under heavy loads, such as testing of a Web site under a range of loads to determine at what point the system's response time degrades or fails
Mutation testing	A method for determining if a set of test data or test cases is useful, by deliberately introducing various code changes ("bugs") and retesting with the original test data/cases to determine if the bugs are detected. Proper implementation requires large computational resources
Orthogonal array testing	Mathematical technique to determine which variations of parameters need to be tested
Pareto analysis	Analyze defect patterns to identify causes and sources
Performance testing	Term often used interchangeably with stress and load testing. Ideally, performance testing (and any other type of testing) is defined in requirements documentation or QA or Test Plans

Test Teknikleri ve Kısa Açıklamaları IV

Positive and negative testing	Testing the positive and negative values for all inputs
Prior defect history testing	Test cases are created or rerun for every defect found in prior tests of the system
Prototyping	General approach to gather data from users by building and demonstrating to them some part of a potential application
Random testing	Technique involving random selection from a specific set of input values where any value is as likely as any other
Range testing	For each input, identifies the range over which the system behavior should be the same
Recovery testing	Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems
Regression testing	Testing a system in light of changes made during a development spiral, debugging, maintenance, or the development of a new release
Risk-based testing	Measures the degree of business risk in a system to improve testing
Run charts	A graphical representation of how a quality characteristic varies with time
Sandwich testing	Integrating modules or programs from the top and bottom simultaneously
Sanity testing	Typically, an initial testing effort to determine if a new software version is performing well enough to accept it for a major testing effort. For example, if the new software is crashing systems every five minutes, bogging down systems to a crawl, or destroying databases, the software may not be in a "sane" enough condition to warrant further testing in its current state
Security testing	Testing how well the system protects against unauthorized internal or external access, willful damage, etc.; may require sophisticated testing techniques

Test Teknikleri ve Kısa Açıklamaları V

State transition testing	Technique in which the states of a system are first identified, and then test cases written to test the triggers causing a transition from one state to another
Statement coverage testing	Every statement in a program is executed at least once
Statistical profile testing	Statistical techniques are used to develop a usage profile of the system that helps define transaction paths, conditions, functions, and data tables
Stress testing	Term often used interchangeably with load and performance testing. Also used to describe such tests as system functional testing while under unusually heavy loads, heavy repetition of certain actions or inputs, input of large numerical values, or large complex queries to a database system
Structured walkthroughs	A technique for conducting a meeting at which project participants examine a work product for errors
Syntax testing	Data-driven technique to test combinations of input syntax
System testing	Black-box type testing that is based on overall requirements specifications; covers all combined parts of a system
Table testing	Testing access, security, and data integrity of table entries
Thread testing	Combining individual units into threads of functionality that together accomplish a function or set of functions
Top-down testing	Integrating modules or programs starting from the top

Test Teknikleri ve Kısa Açıklamaları VI

Unit testing	The most "micro" scale of testing; to test particular functions or code modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. Not always easily done unless the application has a well-designed architecture with tight code; may require developing test driver modules or test harnesses
Usability testing	Testing for "user-friendliness." Clearly, this is subjective, and will depend on the targeted end user or customer. User interviews, surveys, video recording of user sessions, and other techniques can be used. Programmers and testers are usually not appropriate as usability testers
User acceptance testing	Determining if software is satisfactory to an end user or customer
White-box testing	Test cases are defined by examining the logic paths of a system

STATIC TESTING

ProfessionalQA.com™

Concept Of Static Testing

Static testing, a type of software testing methodology, is the verification of a software product, done in a static environment i.e. testing done without executing of the code. It is performed by means of manual and automated reviews of documents. Moreover, it enables early detection of defects during the initial phase of the development cycle.

Features of Static Testing

- It is a stage of **white box testing** and is also referred to as **dry-run testing.**
- Performed manually or through the use of automated tools.
- Improves the quality and ensures the functionality of a software product.
- It does not require the program to be executed.
- Performed during the verification process.
- Detects errors and defects in the early stages of the software development life cycle.

Work document for Static Testing

Static testing involves going through written materials which can give a holistic view of the general working of the product under test. This information can be gained by delving through various documents and reports, other sources like:

- Web page content
- Test Cases
- Design document
- Source code
- Requirements Specifications
- User Document
- Test plans
- Test Scripts

Kaynak: ProfessionalQA.com

Static Testing Techniques

Technical Reviews
An assessment of technical concepts along with a description of suitable alternatives to the product is done.

Inspection
Often used for safety/ time related criticality of the system.

Informal reviews
Involves a general review of working manual of the product & ends with a small bunch of unofficial comments which are never documented.

Walkthroughs
The document of the code is presented to a team of programmers, recorders, and readers.

Static Code Review
Involves checking of syntax, the standards of coding involved, along with the coding optimization.

Merits of Static Testing

- Reduces the cost of rework, as it identifies defects in the early stages of SDLC.
- The feedback received from this testing helps to improve the functioning of the process.
- Offers increased awareness about the various quality issues in the software.
- Improves communication, about critical and important information, among team members.
- Substantially reduces efforts for rework, which further promotes productivity of the development.

Demerits of Static Testing

- The process of static testing can be time consuming, as it is majorly performed manually.
- It inhibits one to find vulnerabilities introduced in the runtime environment.

Things to consider Static Testing

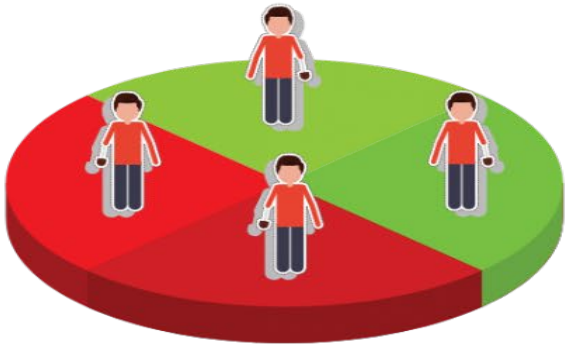
- Thoroughly plan and monitor the review.
- Team should be focused on testing elements.
- Issues and defects should be reported and resolved immediately.
- Team lead or manager should constantly indulge in improvement activities.
- Prioritize the vulnerabilities.
- Tools can be used to enforce secure code practices.

Tools used for Static Testing

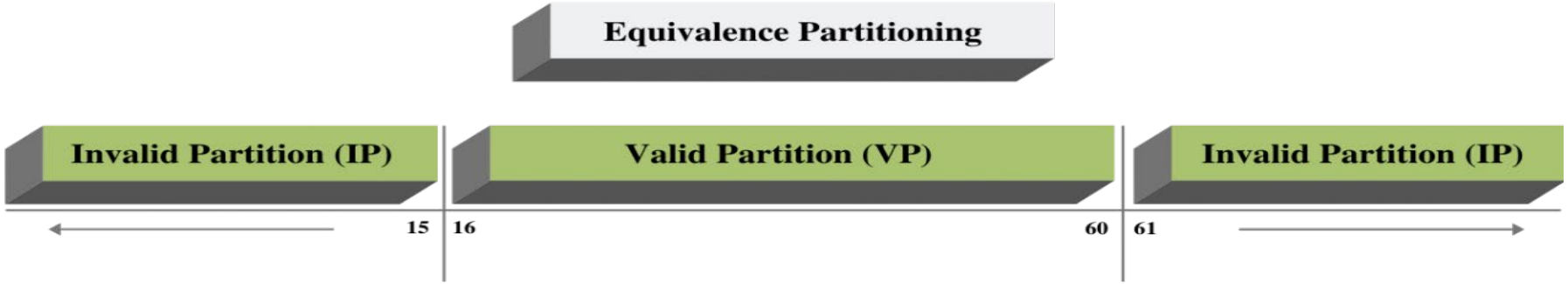
- Veracode.
- RIPS Technologies.
- PVS Studios.
- Kiuwan.
- Gamma.
- Coverity.
- CAST.
- Application Inspector.
- CodeDX.
- IBM Security AppScan.

Black-Box Functional Testing Techniques

- ❑ Equivalence Partitioning
- ❑ Boundary Value Analysis
- ❑ Decision Table Testing
- ❑ State Transition Testing
- ❑ Use Case Testing

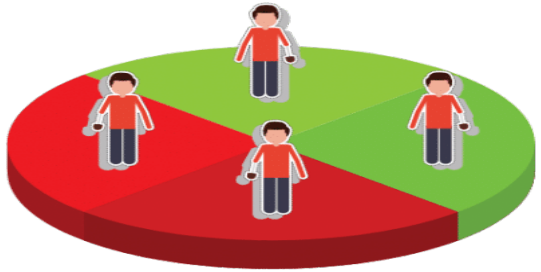


Equivalence Partitioning

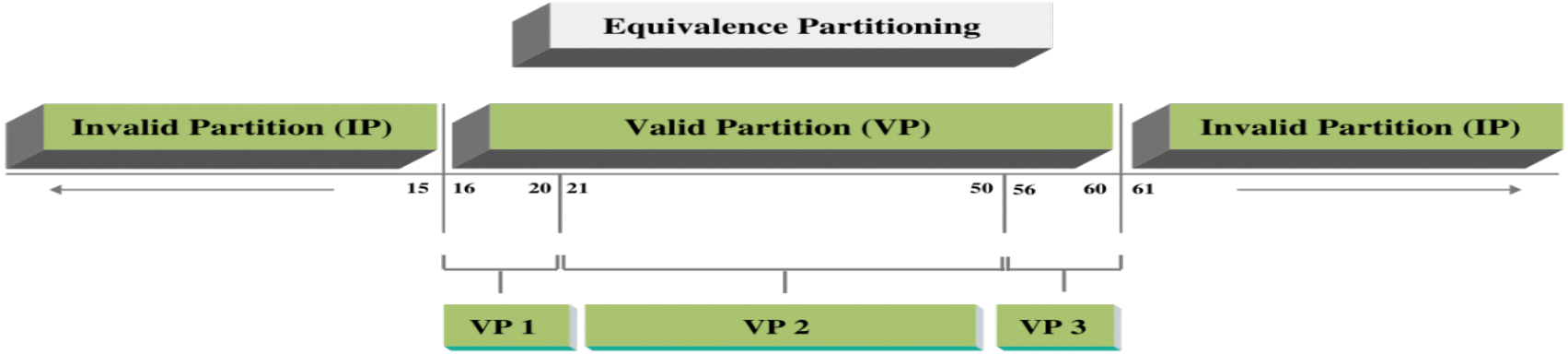


Age = 5
Age = 20
Age = 65 girilir.

Test datası yandaki gibi olduğunda bu verilerle test koşulları %100 sağlanacaktır.



Equivalence Partitioning

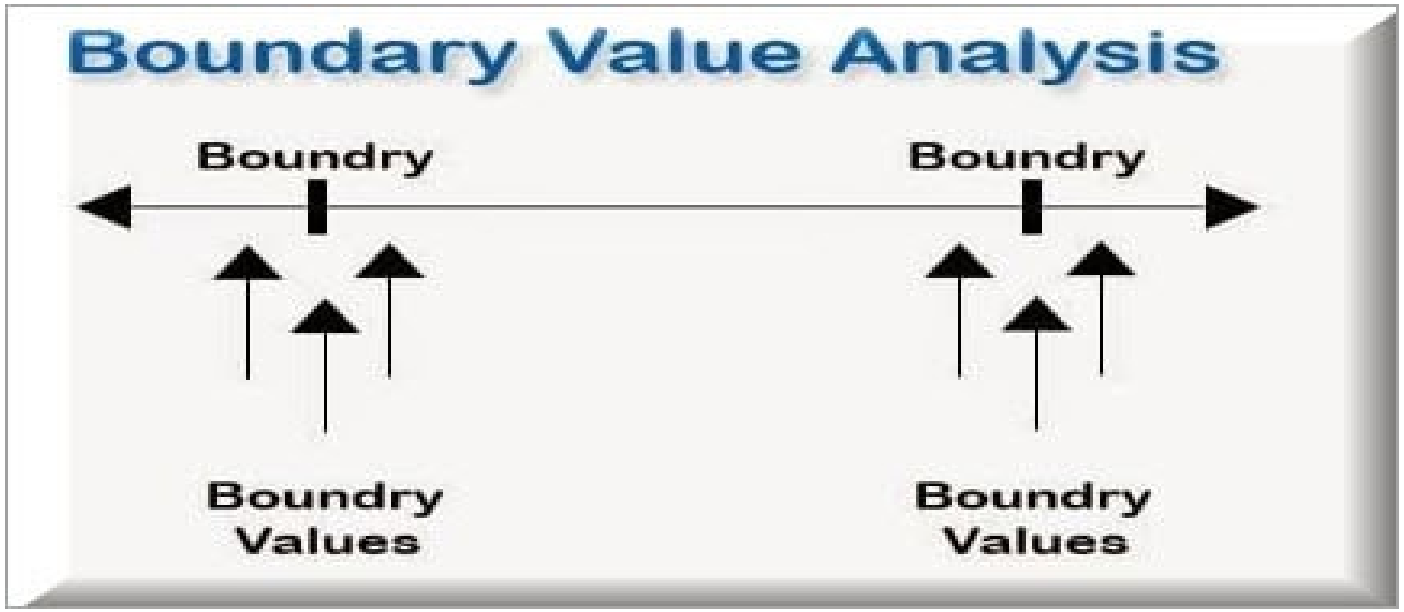


VP1, VP2, VP3 tümü geçerli alt parçalanma olması için test datası
Sıra ile Age = 5, Age = 18, Age = 30, Age = 58, Age = 65 girilir.
Bu beş koşul da tüm gereksinimleri sağlamaktadır. Geçerli (valid) testtir.

Equivalence Class Partitioning (ECP)

AGE Enter Age * Accepts value from 18 to 60

Equivalence Class Partitioning		
Invalid	Valid	Invalid
<=17	18-60	>=61



- Boundary (sınır), sistemin davranışının değiştiği sınıra yakın değerleri ifade eder.
- Sınır değer analizinde, problemi doğrulamak için hem geçerli hem de geçersiz girdiler test edilmektedir.

- 1 den 100 e kadar olan değerleri kabul etmek üzere test etmek istiyoruz.
- Sınır değerleri olarak: 1-1, 1, 1+1, 100-1, 100, ve 100+1 seçebiliriz.
- Bu durumda 1 ‘den 100 ‘e kadar olan değerleri test etmek yerine 0, 1, 2, 99, 100 ve 101 değerleri test edilecektir.

Sistem girişleri

Decision Table Testing

TC ler Test Case

Conditions	Rule 1/TC1	Rule 2/TC2	Rule 3/TC3	Rule 4/TC4
Email Id (Input)	T	T	F	F
Password (Input)	T	F	T	F
Action(Output)	H	E	E	E

Sistem Çıkışı: H ise kullanıcı başarılı giriş yapar, E ise kullanıcı girişi hatalıdır.

Login Ekranının Karar Tablosu ile Testi

Ekranın Güncellenmesi Testinin Karar Tablosu ile Yapılması

Test için gerekenler, girişlerdir (inputs)

Dosya .png formatındadır.

Dosyanın uzunluğu 25 kb. den küçüktür.

Dosyanın çözünürlüğü 132*170px olmalıdır.

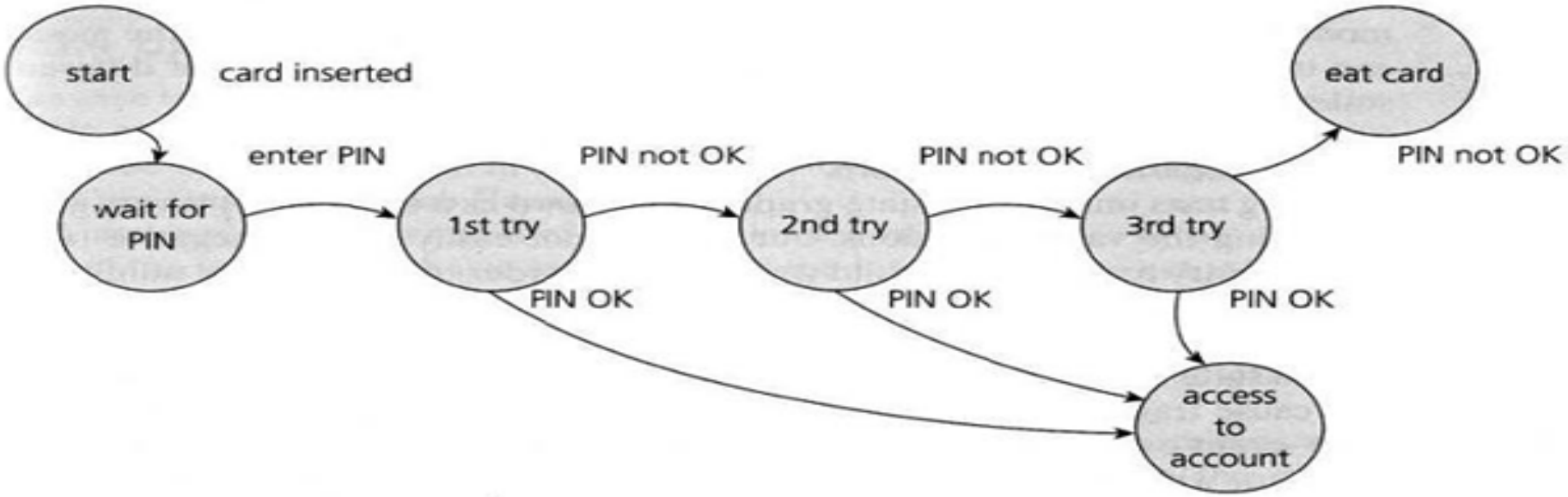
Sistemin çıktısı (output)ise 8 farklı test case ile irdelenir.

Çünkü veri girişi 3 olduğu için 8 farklı test case (TC) tanımlanır (2^3).

Ekran Yüklemesi (Screen Upload) ile İlgili Karar Tablosu

Conditions	Rule 1/TC1	Rule 2/TC2	Rule 3/TC3	Rule 4/TC4	Rule 1/TC5	Rule 2/TC6	Rule 3/TC7	Rule 4/TC8
Format (Input)	.png	.png	.png	.png	Not .png	Not.png	Not.png	Not.png
Size(Input)	<25kb	<25kb	>=25kb	>=25kb	<25kb	<25kb	>=25kb	>=25kb
Resolution (Input)	= 132*170px	!= 132*170px	=132*170px	!= 132*170px	=132*170px	!= 132*170px	=132*170px	!= 132*170px
Action(Output)	Upload the .png file successfully.	Error Message displaying resolution mismatched.	Error message displaying size mismatch	Error message displaying size mismatched	Error Message displaying format mismatched.	Error Message displaying format mismatched.	Error Message displaying format mismatched.	Error Message displaying format mismatched.

State Transition Testing / Black Box Functional Testing



- State Transition testing is a process-oriented test design technique
- State Transition testing focuses on states, events that initiate a transition to another state and actions resulting from such events.
- Tests are designed to execute valid and invalid state transitions.

Use Case Testing

- Sistemin ya da yazılımın fonksiyonel gereksinimlerinin tanımlanması ve sağlanmasına (validation) yardım eder.
- Sistem ya da yazılımın gerçek dünya senaryolarını, kullanıcının gereksinimlerini nasıl sağladığı açıklanır