KTO KARATAY
ÜNİVERSİTESİ

T.C

KTO Karatay Üniversitesi

Fen Bilimleri Enstitüsü

ELEKTRİK BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

TEZLİ YÜKSEK LİSANS PROGRAMI

YAZILIM GÜVENLİK AÇIKLARINA KARŞI GÜVENLİ TASARIM DESENLERİNİN İNCELENMESİ VE UYGULANMASI

Fatih KAPLAN

KONYA

EKİM, 2019

YAZILIM GÜVENLİK AÇIKLARINA KARŞI GÜVENLİ TASARIM DESENLERİNİN İNCELENMESİ VE UYGULANMASI

Fatih KAPLAN

KTO Karatay Üniversitesi Fen Bilimleri Enstitüsü

Elektrik Bilgisayar Mühendisliği Ana Bilim Dalı Yüksek Lisans Programı

Yüksek Lisans Tezi

KONYA

Ekim, 2019

Fen Bilimleri Enstitü Onayı

Prof. Dr. Hüseyin Bekir YILDIZ

Fen Bilimleri Enstitüsü Müdürü

Bu tezli yüksek lisans tezinin yapılması gereken bütün gerekliliklerinin yerine getirdiğini onaylıyorum.

Dr. Öğr. Üyesi Hüseyin Oktay ALTUN

Anabilim Dalı Başkanı

Fatih KAPLAN tarafından hazırlanan "Yazılım Güvenlik Açıklarına Karşı Güvenli Tasarım Desenlerinin İncelenmesi ve Uygulanması" başlıklı bu çalışma 02/10/2019 tarihinde yapılan savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından yüksek lisans tezi olarak kabul edilmiştir.

Dr. Öğr. Üyesi Ali ÖZTÜRK

Tez Danışmanı

Juri Üyeleri

Başkan: Prof. Dr. Novruz ALLAHVERDİ .....

Üye: Doç. Dr. Halife KODAZ .....

Üye: Dr. Öğr. Üyesi Ali Öztürk .....

TEZ BİLDİRİMİ

Tez içerisindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu tezde sunduğum çalışmanın özgün olduğunu bildirir aksi bir durumda aleyhime doğabilecek tüm hak ve kayıplarını kabullendiğimi beyan ederim.

02/10/2019

Fatih KAPLAN

ii

ÖZET

YAZILIM GÜVENLİK AÇIKLARINA KARŞI GÜVENLİ TASARIM DESENLERİNİN İNCELENMESİ VE UYGULANMASI

KAPLAN, Fatih

Yüksek Lisans – Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Dr. Öğr. Üyesi Ali ÖZTÜRK

Ekim 2019

Bu tez çalışmasında, yazılım sistemlerinin geliştirme sırasında ve dağıtım sonrasında ortaya çıkabilecek güvenlik açıklarının hem geliştiriciler hem de son kullanıcılar açısından riskleri incelenmiştir. Güvenlik açıklarının kök nedenlerinin daha iyi anlaşılması açısından sadece entegrasyon ve dağıtım aşamalarında değil yazılım geliştirme aşamasında da meydana gelebilecek güvenlik riskleri ele alınmıştır. Birçok farklı durumda ortaya çıkabilecek güvenlik sorunlarına çözüm tanımlayan güvenli tasarım desenleri açıklanmıştır. Yazılıma sonradan belirli güvenlik mekanizmalarının uygulanmasına odaklanmak yerine, güvenlik açıklarını önlemek üzere yazılım geliştirme aşamasında güvenli yazılım tasarım desenlerinin uygulanması üzerinde durulmuştur. Güvenli tasarım desenleri mevcut en iyi güvenlik tasarım uygulamalarını genelleştirerek ve mevcut tasarım desenlerine güvenlik özellikli işlevsellikleri eklenerek türetilmiştir. Bu çalışmada, güvenli tasarım desenleri örnek bir yazılım üzerinde uygulanarak yazılım geliştirme aşamasında güvenlik açıklarının nasıl önlenebileceği gösterilmiştir.

Anahtar Kelimeler: tasarım desenleri, güvenli tasarım desenleri, güvenli yazılım geliştirme

ABSTRACT

ANALYSIS AND IMPLEMENTATION OF SECURE DESIGN PATTERNS AGAINST SOFTWARE VULNURABILITIES

KAPLAN, Fatih

M.Sc. Electrical and Computer Engineering

Supervisor: Asst. Prof. Dr. Ali ÖZTÜRK

October 2019

In this thesis, the risks of both the developers and the end users were examined in terms of software security vulnerabilities. In order to better understand the root causes of security vulnerabilities, security risks that may occur not only during the integration and deployment phases both also in software development phase, were also discussed. The secure design patterns which describes solutions to various security problems that may occur in many different situations are explained. Instead of focusing on the application of certain security mechanisms after software deployment,this study focused on the implementation of secure design patterns in the development phase to avoid security vulnerabilities. Secure design patterns are derived by generalizing the best available security design applications and adding security-specific functionality to existing design patterns. Furthermore, the implementation of secure design patterns in a sample software application was performed in order to show how to avoid security vulnerabilities in development phase.

Keywords: design patterns, secure design patterns, secure software development

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren akademik danışmanım Dr. Öğr. Üyesi Ali ÖZTÜRK'e kıymetli tecrübelerinden faydalandığım KTO Karatay Üniversitesi Elektrik Bilgisayar Mühendisliği Tezli Yüksek Lisans Bölümü öğretim üyelerine ve Fen Bilimler Enstitüsüne teşekkürü bir borç bilirim.

Fatih KAPLAN

Ekim 2019

İÇİNDEKİLER

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

KISALTMALAR

Kısaltmalar	Açıklama
NFR	Non functional requirement
İOG	İşlevsel olmayan gereklilik
OOP	Object oriented programming
NTP	Nesne tabanlı programlama
UID	User Identifier
GID	Group Identifier
EKL	Erişim kontrol listesi
ACL	Access Control List
CERT	Computer emergency response team
RAII	Resource allocation is initialization
API	Application programming interface
E-TİCARET	Elektronik ticaret
MVC	Model view controller
OWASP	Open web application security project
TCP	Transmission control protokol
SOAP	Simple object access protokol
FTPD	File transfer protokol database
OPENBSD	Open Berkeley software distribution
SSH	Secure Shell
XML	Extensible markup language
RPC	remote procedure call
SCADA	Supervisory control and data acquisition

1. GİRİŞ

1.1 Çalışmanın Amacı

Teknolojinin gelişim hızı, ihtiyaçların çeşitlenmesi ve yazılım geliştiren mühendislerin yeniliklere kolay adaptasyonları sayesinde yazılım geliştirme sektöründe her dönem yeni bir kavram ön plana çıkmıştır. Günümüzün önemli konularından bir tanesi ise yazılım geliştirmede güvenlik konusudur. Büyük sistemlerde açığa çıkan güvenlik zafiyetleri sebebiyle yazılım mühendisliğinde güvenlik konusu önemli bir araştırma alanı haline gelmiştir. Bilgi teknolojilerinin kullanıcıya sunmuş olduğu yapılabilirlik ilkesi sebebiyle ihtiyaçların artması ve günümüzde problemlerin çoğunun programlama ile çözüme kavuşuyor olmasından dolayı modern işletmelerin çok büyük miktarlardaki kurumsal hafıza ve diğer gizli verileri bilgi sistemlerinde depolanmaktadır. Özellikle kişisel veriler başta olmak üzere her türlü verinin gizlilik esası bulunur ve bilginin değerinin güvence altına alınması için yetkisiz her türlü erişime kapalı olması ve izinsiz değişikliklere karşı da korumalı olması gerekmektedir.

Şirketlerin, eğitim kurumlarının, devletlerin ve son kullanıcıların kullanmış olduğu her türlü yazılımda çeşitli veriler tutulmaktadır. Bu verilerin kaybolması veya değiştirilmesi oldukça büyük bir risktir ve mali kayıplara neden olabilmektedir. Verilerin veya sistemlerin korunması için bilgi teknolojileri yöneticileri sisteme kullanıcıların nasıl erişmeleri gerektiğini rol tabanlı olarak veya zaman kısıtlaması ile belirleyerek çeşitli güvenlik politikaları oluştururlar. Fakat bu politika veya dijital poliçeler bilgi sistemleri yöneticileri tarafından kullanışlı olmayan gereklilikler şekline dönüşürse güvenlik için alınan önlemleri içeren politikalar doğru uygulanmamış olacaktır. Mevcut sistemleri güvenlik ilkeleri ve kanunlarda yer alan uygulamalara göre yönetmek olması gereken bilgi sistemleri yönetim şekli olarak kabul edilebilir. Fakat sistemleri bu şekilde yönetmekten çok daha önce yapılması gereken bilgi sistemlerinin geliştirilmesi esnasında güvenlik tasarım desenlerinin benimsenmesi ve yazılım geliştirmede uygulanmasıdır. Güvenlik ile ilgili ortaya çıkan sorunları çözmek için yapı taşlarından bir tanesi yazılım mühendisliğindeki Güvenli Tasarım Desenleridir. Bu desenler, güvenliği tekrar belirleyen genel çözümleri içermektedir. Ek olarak, hali hazırda kullanılmakta olan yazılımların uygulama ve dağıtım aşamalarında güvenlik sorunlarının tespit edilmesi ve giderilmesi ile ilgili pek çok araç

vardır. Fakat güvenlik kusurlarının temel nedenlerinin henüz geliştirme aşamasında iyi bir şekilde anlaşılması güvenlik unsurunun yazılım geliştirmenin tüm aşamalarda göz önünde bulundurulması gibi olumlu bir farkındalık olmasını sağlayacaktır. Yazılım geliştirmede güvenli tasarım desenlerinin uygulanması, problemlerin ele alınabilir alt problemlere dönüştürülebilmesini sağlamaktadır.

Türkiye Cumhuriyeti, Cumhurbaşkanlığı Resmi Gazete' sinde 2019 Temmuz ayında Bilgi ve İletişim Güvenliği Tedbirleri konu başlığı yayınlanan Genelge' de kamu kurum ve kuruluşlarının bilgi ve iletişim güvenliği konusunda uymaları gereken bir takım zorunluluklar açıklanmıştır. Genelge' de yer alan konulardan yazılım geliştirme sürecini ilgilendiren maddeler incelendiğinde yazılımların zararlı kimselerin yapacakları saldırılara karşı güvenli olması gerektiği ve işletim sistemi bazlı kullanıcı ilkelerinin güvenli erişim protokolleri ile sağlanması gerektiğinden bahsedilmiştir.

Bu tez çalışmasında soyutlama seviyesine göre Mimari, Tasarım ve Uygulama seviyesinde 3 ayrı başlık altında toplamda 30'un üzerinde güvenli tasarım deseni incelenmiştir. Ayrıca bu tez çalışmasında güvenli tasarım desenlerinin yazılım geliştirmeye iyi entegre edilmesi ve hiyerarşik olarak güvenlik ile ilgili olan tasarım desenlerinin sınıflandırılarak yöntemlerin geliştirme sürecinde pekiştirilmesi ile ilgili konuları içermektedir. Güvenli tasarım desenlerinin her biri kendi konu başlığında incelenmiş pekiştirilmesi için tasarım deseninin yapısı şekiller ile gösterilmiştir. Güvenli tasarım deseninin kullanımına verilebilecek yaygın örnekler ise şekiller ile gösterilip detaylı bir şekilde açıklanmıştır. Soyutlama seviyesine göre farklı başlıklar altında incelenen güvenli tasarım desenlerinin her bir başlığını ilgilendiren tanımlar literatürdeki ilk Türkçe kaynak olma özelliğini nispeten taşıdığından güvenli tasarım desenlerinin önemi hakkında fikir sahibi olmak isteyenleri oldukça aydınlatacak niteliktedir. Tez çalışmasındaki konuları inceleyen araştırmacılar veya geliştiriciler güvenli tasarım desenleri hakkında teorik bilgiyi edinmiş olacaklardır. Konuların içerisinde yer alan şekiller ile ilgili güvenli tasarım deseninin ne şekilde çalıştığı yorumu araştırmacı veya geliştirici tarafından yorumlanabilecektir. Soyut konulardan örnek şekiller ile biraz daha somut düşünmeye başlayacak olan araştırmacı veya geliştiricileri dördüncü bölümde örnek bir uygulama beklemektedir. Örnek uygulama ile tez çalışması içerisinde incelenen güvenli tasarım desenlerinden bazıları ele alınmış

örnek kod ve ekran görüntüleri ilgili bölümde gösterilmiştir. Güvenli tasarım desenlerinin uygulanabileceği geniş alanlardan bir tanesi e-ticaret ve güvenli ödeme sistemleridir. Örnek uygulamada e-ticaret sistemleri üzerinde meydana gelebilecek zafiyetlere karşı kullanılmasında fayda olan güvenli tasarım desenleri incelenmiş sonuç bölümünde ise farkları ve yararlarına değinilmiştir. Sonuç bölümünde, genel olarak güvenli tasarım desenlerinin kullanımının ne gibi faydalar sağladığına ve örnek uygulamada ne gibi yararlar sağladığı ile ilgili bilgiler açıklanmıştır. Tez çalışmasının sonucundan en iyi şekilde istifade eden bir araştırmacı veya geliştirici güvenli tasarım desenlerinin kullanımına ilişkin teorik ve pratik bilgiye sahip olacaktır.

1.2 Tasarım Desenleri

Yazılım tasarım desenleri, başka bir deyişle yazılım tasarım şablonları, yazılım geliştirmede düzenli olarak ortaya çıkan sorunlara karşı geliştirilmiş çözümler bütünüdür. Yazılım geliştirme süreci için kolayca uygulanabilir, zamandan tasarruf edilmesine olanak sağlayan stratejiler sunmaları amaçlanmıştır. Düzgün bir şekilde kurgulanmış ve tanımlanmış bir tasarım desenine eşlik eden bir dokümantasyon ile geliştiriciler verilen problemi hızlı bir şekilde belirleyebilir ve uygulayabilirler.

İlk tasarım deseni notasyonu Christopher Alexander tarafından uzay ve yapı mimarisi için yeniden kullanılabilir stratejiler konusundaki çalışmalarında geliştirilmiştir. Beck ve Cunningham her desenin çevremizde tekrar tekrar ortaya çıkan bir problemi anlattığına değinmişlerdir [1]. 1987' de Beck ve Cunningham Alexander'ın desen fikrini, nesne yönelimli bir programlama dili olan Smalltalk' taki grafiksel kullanıcı ara yüzlerinin geliştirilmesinde uygulamışlardır. Cunningham ve Beck, bir Smalltalk kullanıcı ara yüzü geliştirmek için çeşitli modeller geliştirerek elde etmiş oldukları 23 farklı sonucu OOPSLA-87 atölyesinde yayınlamışlardır. 1990-1992 yılları arasında ise Gamma ve ark. dörtlü çete olarak bilinen GoF desen kataloğunu derlemişlerdir. Bu modeller daha sonra Gang of the Four (GoF) kitabı olarak bilinen yazılım tasarım kalıplarının ilk son derece sezgisel kitabını yayınladılar [2]. Bu kitabı günümüzde yaklaşık 2,000 kalıptan oluşan Grady Booch'un Yazılım Mimarisi El Kitabı gibi birçok desen koleksiyonunun yayınlanması izlemiştir [3].

1.3 Güvenli Yazılım Geliştirme

Yazılımda güvenliği, bir kurumun verilerinin istenmeyen erişime veya değiştirilmeye karşı korunması olarak ifade edebiliriz. Barbacci ve ark. Yazılım mühendisliği enstitüsünde belirtmiş oldukları gibi güvenlik, kaçak kullanıcı girdisinden hassas bilgilere ve yetkisiz erişime karşı korumaya kadar değişen uygulamaları içerir [4].

Güvenlik, geleneksel olarak bir sistemin nasıl çalışması gerektiğine dair işlevsel olmayan gereksinimlerden biridir. Güvenlik, ne yazık ki genellikle tasarım sürecinin sonlarında veya dağıtım aşamasından sonra düşünülür [5]. Bu bir kurumun verilerini risk altına sokma açısından bakımı daha zor ve daha az güvenli bir sistem olmasına yol açar.

İşlevsel olmayan gereksinimler çoğu zaman sistem geliştirilirken belirtilir, çünkü gereksinimleri standart olarak formüle etmek zorlaşmaktadır. Burada bahsedilen güvenlikle ilgili İşlevsel Olmayan Gereksinim (İOG)' lerin yönetilmesi stratejisi ilk önce güvenliği daha spesifik İOG' lerin hiyerarşisine indirgemektedir. Daha sonra ayrıştırılmış gereklilikleri, güvenlik gelişimi için en iyi uygulamalar kataloğu ile ilişkilendirmektedir. İOG güvenliğinin ayrışması iki adımda gerçekleştirilmiştir. Chung ve ark. İOG ayrıştırma üzerindeki çalışması güvenliği seviyeler şeklinde özelliklere ayırmak için kullanılmıştır [5]. Birincisi gizlilik; bir işletmenin hassas verilerinin yetkisiz erişime karşı korunması, ikincisi bütünlük; kuruluşun verilerinin yetkisi değiştirilmeden korunmasıdır. Üçüncüsü erişilebilirlik; bir kuruluşun verilerinin yetkili kullanıcılar tarafından kullanılamamasına karşı korunmasıdır. Dördüncüsü; hesap verebilirlik: işletmenin verilerini etkileyen eylemleri gerçekleştiren kişi veya kişilerle olan birliğidir.

İOG kavramının etkinliğinin belgelenmesi için Chung ve diğer geliştiriciler tarafından bir çerçeve çalışması yapılmıştır ve bu çalışmada her bir İOG' nin belirli bir konu ile ilişkilendirilmesi amaçlanmıştır. Bu çalışmaya örnek vermek gerekirse yetkisiz bir erişime karşı koruma gerektiren hesap veya belirli bir süre içerisinde yanıt vermesi gereken bir servis düşünülebilir. İOG' leri ayırmak için kullanılan bu çerçeve güvenliği dört spesifik başlık ile ele almıştır. Bu başlıklar yukarıda da bahsedildiği gibi: gizlilik, bütünlük, erişilebilirlik ve hesap verebilirlik şeklindedir.

Gizlilik ile ilgili gereklilikler, bir sistemin nasıl tasarlandığını, erişimin nasıl kontrol edildiğini ve süreçlerin nasıl kontrol edildiği ile kullanıcıların nasıl doğrulandığını düşünmektedir. Gizliliğin ihlali, genellikle kötü niyetli taraflarca tehlikeye atılan verilere yetkisiz erişim sağlanmak için istismar edilen sistemlerdeki hatalara bağlanabilir. Günümüzde verilerin gizliliği ve güvenliği ile ilgili alınan önlemler, yazılım geliştirmede güvenli tasarım desenlerinin etkin kullanımı ile büyük ölçüde önlem niteliği taşımasının yanı sıra hükümetlerin artık bilgi güvenliği yönetim sistemi ve veri güvenliği politikaları ile kişisel verilerin korunması kanunun üzerinde yürütmekte oldukları çalışmalar ile önemli bir yer almıştır.

Güvenli tasarım desenleri soyutlama seviyelerine göre Mimari, Tasarım ve Uygulama olarak sınıflandırılmaktadır. Tasarım deseni kavramındaki tasarım bir sorun için belirlenmiş olan genel bir çözümdür ve yeniden kullanılabilir bir yapıdadır. Tasarım kavramı, algoritma veya doğrudan koda dönüşebilen bitmiş bir tasarım değildir. Algoritmalar hesaplama problemlerine odaklanır ve onları çözerler. Tasarımlar ise pek çok farklı durumda kullanılabilecek bir problemin nasıl çözüleceği ile ilgili açıklama niteliğindeki şablonları ifade eder. Güvenli tasarım desenleri, yazılım geliştirmede güvenlik açıklarının yanlışlıkla geliştirilen projeye eklenmesini ortadan kaldırmak ve olası güvenlik açıklarının sonuçlarının etkisini azaltmak için kullanılmalıdır. Güvenli tasarım desenleri/kalıpları erişim kontrolü gibi güvenlik mekanizmalarını tanımlayan yazılım tasarım kalıplarıdır. Güvenli desenler teriminin ilk kullanımı 1997’ de Yoder ve Barcalow tarafından yazılmıştır [6].

Birinci bölümde çalışmanın amacı, tasarım desenleri ve güvenli tasarım desenleri ile ilgili bilgiler ve güvenli yazılım geliştirme süreçlerinden bahsedilmiştir.

İkinci bölümde yapılmış çalışmalara dikkat çekilerek literatür özetine yer verilmiştir ve bu bölümde incelenen literatür çalışmalarında güvenli tasarım desenlerinin yazılım güvenlik açıklarına karşı incelenmesi konularının üzerinde çok durulmadığı görülmüştür.

Üçüncü bölümde güvenli tasarım desenlerinden literatürde yer alan ve en yaygın olanları incelenmiş olup tek bir kaynakta incelenen ilk Türkçe güvenli tasarım deseni rehberi niteliğinde her bir desen açıklanmıştır.

Dördüncü bölümde, üçüncü bölümde yer alan yöntemlerin uygulanabilmesi için örnek bir yazılım projesi geliştirilmiştir. Örnek projede üçüncü bölümde yer verilen metotlardan bazıları uygulanarak ekran görüntüleri ile pekiştirilmiştir.

Beşinci bölümde ise geliştirilen yöntemlerin uygulama üzerindeki faydaları ve avantajları tablo halinde verilmiştir. Beşinci bölümde ayrıca üçüncü bölümde anlatılan güvenli tasarım desenlerinin ileride yapılabilecek çalışmalarda ne ölçüde geliştirilebileceği ve ne tür iyileştirmeler yapılabileceğinden bahsedilmiştir.

2. LİTERATÜR ÖZETİ

Güvenli tasarım desenleri, yazılımda tekrar eden güvenlik sorunlarını çözmek için kullanılan modül ve program parçalarıdır. Standart tasarım desenlerinden farklı olarak tekrar etme olasılığı olan ve kullanılması sakıncalı olan kodları bir disiplin ile geliştirmek için tekrar kullanılabilen birden fazla sınıftan oluşmuş modül ve program parçalarını içerir. Her tasarım deseninin bir adı vardır ve hangi amaçla kullanılacağı açıklanmıştır. Tasarım desenleri kullanılarak geliştirilen yazılımlarda, yazılım ekibinde tasarım desenlerini tanımayan geliştiriciler için ilgili tasarım desenini öğrenmeye ve kullanmaya yönlendirecek bir motivasyon kaynağı oluşturur.

Tasarım desenlerinin kullanılması kavramsal olarak daha üst seviyede çalışılmasını ve düşünülmesini sağlar. Nesneler seviyesinde sorunları çözmek her zaman kolay olmayabilir, fakat tasarım desenleri sayesinde problem çözmek kolaylaşmaktadır. Yazılım geliştirme sürecinde önemli olan husus sadece yazılımın düzenli bir standartta ve disiplinde geliştirilmesi ve diğer programcılar tarafından anlaşılır olması değildir. Geliştirilen yazılımın güvenli olması da gerekmektedir. Güvenli yazılım geliştirme sürecinde de uyulması gereken birtakım standartlar güvenli tasarım desenleri konu başlığı ile incelenmektedir. Bu çalışmada standart kabul edilen güvenli tasarım desenleri incelenmiş ve örnek programlar ile sonuçlar ortaya konulmuştur.

Yazılım geliştirme stratejileri, gerçekleştirilebilir gereksinimlerin geliştirilmesinin zorluğu, uygun tekniklerin belirlenmesi, uygulanması ve güvenli tasarımın öğretilmesi nedeniyle genellikle güvenlikle ilgili hususları göz ardı etmektedir. Bu çalışma, bu endişeleri ele almak için çeşitli stratejileri açıklamaktadır.

Literatürde Chung ve ark. çalışmalarına dayanan sistem güvenliği iki seviyeli bir karakterizasyondan türetilen kesin gereksinimleri ortaya çıkarmak için ayrıntılı sorular sağlamaktadır [7]. Güvenli yazılım geliştirme için bu karakterizasyonu daha önce yayınlanmış stratejilere veya desenlere bağlamak için yeni bir çerçeve kullanmaktadır. Dangler' in çalışmasında ise güvenli yazılım geliştirme için güvenli yazılım geliştirme desenleri karakterizasyonunu daha önce yayınlanmış stratejilere veya desenlere bağlamak için yeni bir çerçeve kullanılmaktadır [8]. Dahil edilen örnek olaylar, güvenli tasarım yani sınırlı bir görünüm, rol tabanlı erişim kontrolü, güvenli durum

makinesi gibi diğer modellerin belge yönetimi için bir üretim sistemine uygulanmasını içeren çerçevenin etkinliğini göstermektedir.

Ülkemizde yapılan çalışmalar incelendiğinde Tubitak-Bilgem tarafından hazırlanan güvenli yazılım geliştirme el kitabında güvenli yazılım geliştirme tasarım desenleri incelenmemiştir [9]. Fakat siber güvenlik enstitüsü; verinin korunması, kimlik doğrulama, yetkilendirme, erişilebilirlik, izleme, denetim ve çeşitli güvenlik önlemleri başlıkları altında konuyu incelemiştir. Tez çalışmasında öncelikle güvenli tasarım desenleri incelenmiş ve yazılım geliştirme sürecinde uyulması gereken güvenlik adımlarına da değinilmiştir. Bu bağlamda Tubitak-Bilgem siber güvenlik enstitüsü tarafından hazırlanmış olan dokümanda olduğu gibi güvenlik adına detaylı bir çalışma yapılmış olup Tubitak dokümanından farklı olarak ise bu çalışmaların bir standart dahilinde güvenli tasarım desenleri çerçevesinde incelenmesi amaçlanmıştır.

Ülkemizde yapılmış akademik çalışmalar incelendiğinde ise tam olarak güvenli yazılım geliştirme tasarım desenleri üzerinde yapılmış bir çalışma olmasa da benzer bir konu Gazi Üniversitesi Bilgisayar Mühendisliği öğrencileri tarafından incelenmiştir. Yapılan çalışmada web uygulama güvenliği açıkları ve güvenlik çözümleri üzerine bir araştırma yapmışlardır. Aydoğdu ve Gündüz OWASP tarafından yayınlanan ilk 10 web uygulama güvenlik açıkları, açıkların kaynakları ve bu açıkları istismar eden saldırıları önlemek için kullanılan güvenlik çözümleri araştırılmış, bu açıkları kullanarak gerçekleştirilebilecek saldırılara karşı alınabilecek önlemler, kullanım alanları, platform bağımsızlığı, çalışma mantığı ve verimlilikleri açısından değerlendirilerek karşılaştırılmıştır [10]. Elde edilen bilgiler ve bulgular doğrultusunda, hangi tür saldırılara karşı nasıl bir güvenlik çözümünün alınması, tercih edilmesi konusunda öneriler, farkındalığın ve web uygulama güvenliğinin arttırılmasına yönelik çözümler incelenmiştir. Literatür taraması kapsamında incelenen bu makale yalnızca web uygulama güvenliği açıkları üzerinde çalışılmış olduğundan web uygulamaları dışında geliştirilen yazılımlarda bir önlem niteliğinde değildir. Ayrıca yalnızca açıklar üzerinde durulduğundan tez çalışmasında amaçlanan güvenli yazılım nasıl geliştirilir, standartlar nelerdir gibi sorulara da karşılık vermemektedir.

Güvenli tasarım desenlerinin daha özel bir alan olarak e-ticaret sistemlerinde incelenerek güvenli tasarım ile ortaya çıkan NFR çerçevesini kullanan bir çalışmada ise güvenlik adımının geliştirmeden sonra düşünülmemesi gereken önemli bir adım olduğuna değinilmiştir [11]. Towson Üniversitesi öğrencilerinin yapmış oldukları bu çalışmada NFR yaklaşımı e-ticaret sistemleri üzerinde incelenmiştir. Çalışmada geliştirme sürecinden sonra düşünülen güvenlik süreci çok daha maliyetli bir süreç olacağından bahsedilmiştir. Yine yaptıkları çalışmada son yıllarda birçok deneyimli araştırmacının daha iyi ve daha ucuz yazılımlar için tasarım desenlerini keşfetmeye ve kategorilendirmeye büyük çaba sarf ettiklerini, bilgi birikimi ve deneyim ile güvenlik konusuna odaklanan tasarım modellerinin ortaya çıktığından bahsedilmiştir. Bu makalede bahsedilen NFR yaklaşımı İngilizce kelime karşılığı literatürde non-functional requirements olarak geçmektedir ve Türkçe olarak işlevsel olmayan gereksinimler anlamına gelmektedir. Towson Üniversitesi öğrencilerin e-ticaret sistemleri üzerinde özelleştirerek inceledikleri güvenli tasarım desenleri makalesinde Wang ve ark. tasarım örüntülerinin doğası gereği çok sayıda tekrar eden problem ve çözüm için uygulanabilecek yüksek seviyeli açıklamalar nedeniyle bazen sistemin hangi yönünün gerçekten daha güvenli olduğunun belirlenmesinin zor olduğundan bahsedilir ve performans ile kullanılabilirlik gibi diğer NFR (işlevsel olmayan gereksinimler) için herhangi bir endişe duymadan sadece güvenliğin düşünülmek istenmediğini ifade etmektedir [11]. Dolayısıyla diğer NFR'leri dikkate alarak güvenlik tasarım desenlerinin objektif bir analizini yapmak ve tüm NFR'ler arasında dengeleri ve çelişkileri sistematik olarak temsil etmek gerektiği üzerinde örnek bir e-ticaret uygulaması ile çalışma yapmışlardır.

Tez çalışmamıza yakın bulduğumuz yapılmış bir diğer çalışma ise güvenlik desenleri isimli bir araştırma olan ve Florida Atlantic Üniversitesi tarafından yapılmış olan çalışmadır. Fernandez ve ark. bu çalışmada öncelikle doğrulama, yetkilendirme, rol tabanlı erişim denetimi, güvenlik duvarları, web hizmetleri güvenliği gibi konuları incelemesi ile Tubitak Bilgem tarafından hazırlanan güvenli yazılım geliştirme temel dokümanına benzemektedir [12]. Farklı olarak incelenen başlıklar tek tek açıklanmış olup örnek programlar ile pekiştirilmiştir. Ayrıca bu çalışma UML modellerini kullanmış olması açısından tez çalışması ile benzerlik göstermektedir. İncelenen diğer

çalışmalarda da olduğu gibi bu tez çalışmasında istenen araştırma ve sonuçlar tam olarak yapılmamıştır. Florida Atlantik Üniversitesi'nden Eduardo B. ve arkadaşlarının yapmış oldukları bir çalışma ise güvenlik modellerini kullanarak güvenli scada sistemlerini tasarlamayı konu almaktadır. Scada, merkezi denetleme kontrol ve veri toplama olarak bilinen bir sistemdir. Yaptıkları çalışmada şehirlerdeki elektrik veya su dağıtım şirketleri gibi altyapı sistemlerinin günlük yaşantımızı destekleyen sistemler olduğu için bu sistemlerde çalışan yazılımların korunmaları gerektiğine değinilmiştir. Sistemleri analiz etmek, inşa etmek ve değerlendirmek için kullanılan güvenlik desenlerinin scada sisteminde kullanılarak güvenli bir scada sistemi kurulmasında çeşitli yöntemlerini önermişlerdir. Çalışmalarında genel bir scada sisteminin mimarisini incelemiş ve ona yönelik olası saldırıları analiz ederek ardından bu saldırılara karşı dayanıklı bir scada sistemi tasarlanması için güvenli tasarım desenlerini kullanmışlardır. Güvenli tasarım desenlerini güvenli scada sistemleri için araştırmak güvenli tasarım desenleri konusu hakkında yeni bir yön olarak değerlendirilmektedir [13]. Waikato üniversitesi'nden Steve Reeves ve ark. yaptıkları bir çalışmada ise güvenli uygulamalar geliştirmek için güvenli platformlar, güvenlik modelleri ve platform özelliklerini uygulamak için güvenli tasarım desenlerinin uygulanması gerektiğine değinmişlerdir. Güvenli tasarım desenlerini ve platform özelliklerini paketleyen ve güvenlik özellikleriyle ilgili güvence sağlamak için doğrulanan yeniden kullanılabilir doğrulanmış tasarım desenleri konseptini önermişlerdir [14]. Florida Atlantik Üniversitesi'nden Eduardo B. ve ark. bir başka çalışmaları ise güvenli tasarım desenleri kullanılarak güvenli bulut mimarileri oluşturma konu başlığı ile ele alınmıştır. Yapılan çalışmada halen büyümekte olan yüzden fazla güvenlik modelinin bir kataloğunu oluşturmuşlar ve bu modelleri, bir saldırganın saldırgan bakış açısından nasıl gerçekleştirildiğini ve nasıl durdurulabileceğini açıklamışlardır [15]. Siegen Üniversitesi'nden Obaid Ur ve ark. akıllı ölçüm sistemlerinde güvenlik için güvenli tasarım desenleri başlıklı çalışmalarında akıllı ölçüm için güvenli tasarım modellerinin kullanımından bahsetmişlerdir. Akıllı ölçüm sistemlerinde güvenlik açıklarından büyük ölçüde yararlanılabileceği endişeleri bulunduğundan bireysel düzeyde servis sağlayıcılara yanlış ölçüm verisi sağlamak gibi problemlerin önüne geçmek için güvenli tasarım desenlerinin uygulanması gerektiğini vurgulamışlardır. Yapılan çalışmada akıllı

ölçüm sistemlerine olası güvenlik saldırılarının bir listesini tanımlamış ve özetlemişlerdir. Güvenli tasarım desenlerinin bir sistemdeki tasarım hatalarını azalttığından bahsederek akıllı ölçüm sistemlerinde güvenli yazılım geliştirmek için güvenlik modeli tabanlı bir mühendislik yaklaşımı tanıtılmıştır [16].

Bu tez çalışmasında öncelikle güvenli tasarım desenlerinin her birinin nasıl ortaya çıktığından, ne şekilde kullanılması gerektiğine, kullanımını doğuran sebeplere ve doğru kullanımına değinilmiş literatürdeki güvenli tasarım desenleri konusunu ele alan tek tez çalışması haline gelmiştir. Güvenli tasarım desenleri ile ilgili gereken bilgi ve örnekler verildikten sonra yapılmış çalışmalarda da incelenen örneklerde olduğu gibi rol tabanlı, uygulama tabanlı, sunucu taraflı, kimlik doğrulama gibi güvenliğin aksatılmaması gereken noktalarına değinilerek örnek uygulamalar ile sonuçlar ortaya konulmuştur. Güvenlik modellerinin ve yazılım geliştirmedeki değerlerin farkında olması için yapılan bu çalışmada giriş doğrulama işlemleri, tamsayı taşması, giriş doğrulama, HTTPS, dosya erişimi ve SQL enjeksiyonu konuları da ele alınmış ve son olarak sonuç ortaya konularak konunun önemi belirtilmiştir.

3. GÜVENLİ TASARIM DESENLERİ

Güvenli tasarım desenleri, güvenlik açıklarının gerçekleştirilen kodlamaya girmesini önlemek veya güvenlik açıklarının sonuçlarının etkisini azaltmak için genel bir tasarım rehberliği sunar. Güvenli tasarım desenleri, nesne yönelimli tasarım yaklaşımlarıyla sınırlı değildir. Prosedürel programlama dillerine de uygulanabileceğini söylenebilir. Bu modeller güvenli kodlama kılavuzlarından daha yüksek bir soyutlama seviyesindedir. Güvenli tasarım desenleri, erişim kontrolü, kimlik doğrulama ve yetkilendirme gibi belirli güvenlik mekanizmalarını tanımlamak yerine güvenli geliştirme süreçlerini tanımlar ve mevcut güvenlik sistemlerinin yapılandırılması hakkında rehberlik sağlaması sebebi ile güvenlik modellerinden farklıdır.

3.1 Mimari Seviye Güvenli Tasarım Desenleri

Mimari seviye desenler, sistemin farklı bileşenleri arasında üst düzey sorumluluk dağılımına odaklanır ve bu üst seviye bileşenler arasındaki etkileşimi tanımlar. Mimari seviye desenleri:

- Güvenilmez Ayrışma Tasarım Deseni (Distrustful Decomposition Design Pattern)
- Ayrıcalık Ayrıştırma Tasarım Deseni (Privilege Seperation Design Pattern)
- Çekirdeğe Erteleme Tasarım Deseni (Defer To Kernel Design Pattern)
- Hatalarla Tam Görünüm Tasarım Deseni (Full View With Errors Design Pattern)

3.1.1 Güvenilmez Ayrışma Tasarım Deseni

Güvenli tasarım deseninin amacı, ayrı işlevleri karşılıklı güvenilmeyen programlara taşımaktır.

Birçok saldırı yüksek izinlerle çalışan hassas uygulamaları hedeflemektedir. Bu durum saldırganın fazla bilgiye erişmesine izin vermekte veya saldırganın uygulamada daha kısıtlayıcı izinlerle çalıştığından daha fazla güvenlik açığı kullandıktan sonra daha fazla zarar vermesine neden olmaktadır. Bu tür saldırı sınıfının bazı örnekleri aşağıdaki gibidir.

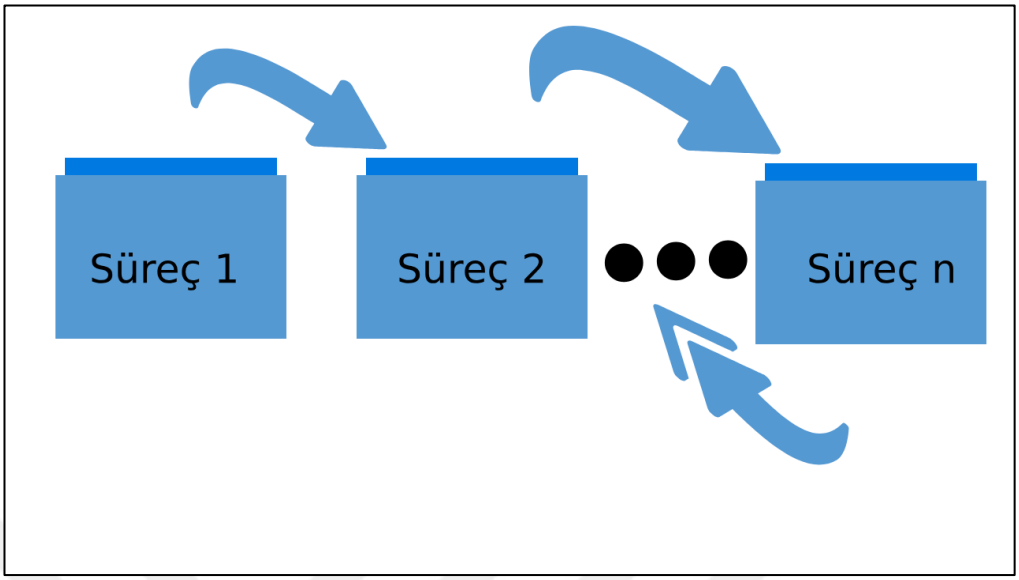
- Internet Explorer'ın yönetici ayrıcalıklarına sahip bir hesapta çalıştırıldığı çeşitli saldırılar,

- Saldırganın rastgele kodu kök (root) olarak çalıştırmasına izin veren telnet servis dosyalarında arabellek taşması sonucu meydana gelen güvenlik açığı,
- Norton Anti Virüs firmasının 2005 yılında saldırganların yönetici ayrıcalıklarıyla çalışırken keyfi senaryo (script) komut dosyaları çalıştırmalarına izin veren güvenlik açıkları

Bu saldırıların tümü, UNIX altındaki Kök kullanıcı veya Windows altındaki yönetici yetkisine yükseltilmiş ayrıcalıklarla çalışan bir uygulamada bulunan güvenlik kusurlarından yararlanmaktadır ve ardından bilgisayarda çalışan diğer uygulamaları tehlikeye atmak için uygulamanın yükseltilmiş ayrıcalıklarını ve temel işlevlerini hassas verilere ulaşmak için kullanır. Güvenilmez ayrışma tasarım deseni, güvenlik açıklarını sistemin küçük bir kümesine indirgeyerek sistemin tek bir bileşenini tehlikeye atarak sistemin tümünün tehlikeye girmesini engeller.

Bu sayede kötü niyetli faaliyetlerde bulunan saldırganların elinde, tüm uygulamanın işlevselliğini ve verilerini değil yalnızca tehlikeye atılmış bileşenin işlevselliğine ve verilerine sahip olacaktır. Bir sistemin güvenilmez ayrışma tasarım desenini kullanabilmesi sistemin birden fazla üst düzey işlev gerçekleştirebildiği ve çeşitli işlevlerinin farklı ayrıcalık seviyesine sahip olması gerektiği durumlarda yararlı olacaktır.

Güvenilmez ayrışma (Distrustful Decomposition) tasarım deseni ayrıcalık azaltma (Privilege Reduction) olarak da bilinmektedir [17]. Bu modelin genel yapısı, sistemi her biri potansiyel olarak farklı ayrıcalıklara sahip olacak şekilde ayrı işlemler olarak çalışan iki veya daha fazla programa böler. Her işlem, sistem işlevselliğinin küçük ve iyi tanımlanmış olan bir alt kümesini işler. İşlemler arasındaki iletişim, soketler, RPC, SOAP veya paylaşılan dosyalar gibi işlemler arası bir iletişim mekanizması kullanarak gerçekleşir. Şekil 3.1'de güvenilmez ayrışma tasarım deseninin yapısı gösterilmektedir.



Şekil 3.1. Güvenilmez Ayrışma Tasarım Deseninin Yapısı

Güvenilmez ayrışma, tek bir bileşen programın başarıyla kullanılması durumunda saldırganın tüm sistemi tehlikeye atmasını önler. Çünkü başka bir program, tehlikeye giren programın sonuçlarına güvenmeyecektir.

Uygulamada bu desen aslında işletim sisteminde zaten bulunan standart işlem / ayrıcalık modelinin ötesinde bir şey kullanmaz. Her program potansiyel olarak farklı kullanıcı ayrıcalıklarına sahip kendi süreç alanında çalışır. Farklı programlar arasındaki iletişim, tek yönlü veya iki yönlü gerçekleşir. Tek yönlü iletişim kontrolünde, kontrolü çalıştırmak için veya süreci oluşturmak için bazı programlama metotları (fonksiyon, metot) kullanılabilir veya işletim sistemine özgü başka bir yöntem kullanılır. İki yönlü iletişim kontrolünde TCP veya SOAP gibi iki yönlü süreçler arası iletişim mekanizması kullanılır.

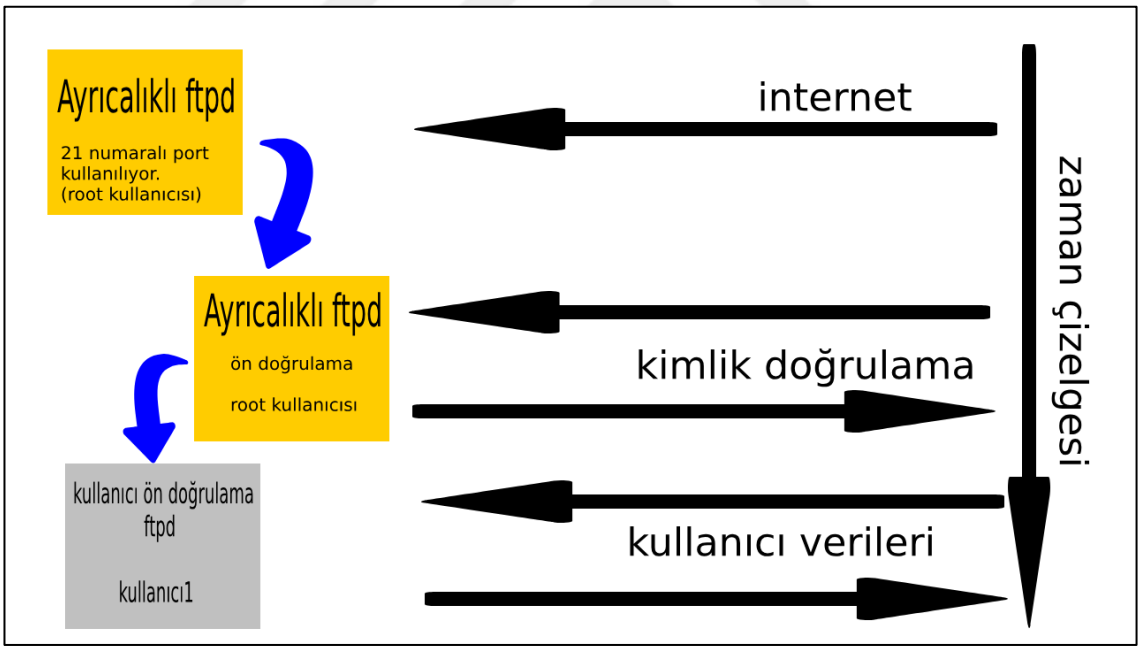
Güvenilmez ayrışma tasarım deseni ile ilgili bilinen uygulamalar aşağıdaki gibidir.

- Qmail posta sistemi
- Postfix posta sistemi
- Microsoft uygulama yönetici ayrıcalıkları
- Microsoft Windows Vista kullanıcı hesabı kontrolü

Bu desenin uygulandığı en iyi örnek kullanıcılar, sistemler ve yazılım bileşenleri arasındaki etkileşimin geniş bir kombinasyonunu içeren karmaşık bir sistem olan Qmail posta sistemidir.

3.1.2 Ayrıcalık Ayrıştırma Tasarım Deseni

Bu desenin (Privilege Separation) amacı programın işlevselliğini etkilemeden veya sınırlamadan, özel ayrıcalıklı kod miktarını azaltmaktır. Bu desen güvensiz ayrışma deseninin daha spesifik bir örneğidir. Birçok uygulamada basit bir işlem kümesi daha yüksek ayrıcalıkları gerektirmektedir. Buna karşın çok daha büyük karmaşık ve güvenlik hatalarına açık işlem kümesi de normal ayrıcalıklı bir kullanıcı bağlamında çalıştırılabilir. Şekil 3.2'de ayrıcalık ayrıştırma deseninin uygulanabileceği bir FTPD sistemi ve bu desenin kullanılmaması durumunda ortaya çıkabilecek güvenlik sorunlarının ayrıntılı bir görünümü yer almaktadır.



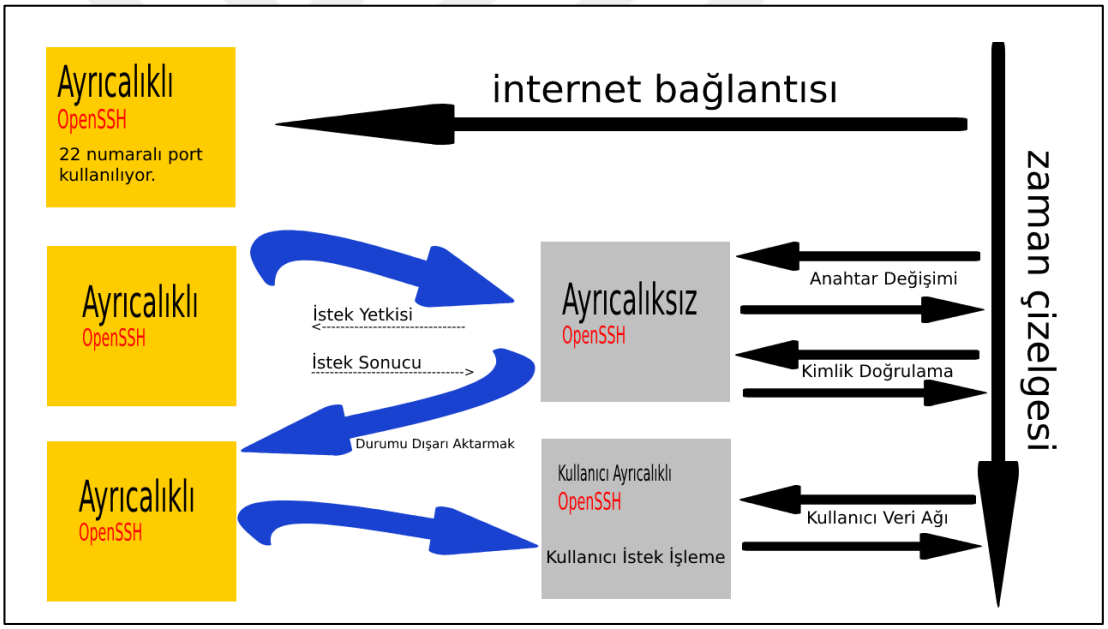
Şekil 3.2. Ayrıcalık Ayrıştırma Deseninin Uygulandığı FTPD Sistemi

Güvenlik açığı ayrıcalıklı sunucusu henüz güvenilmeyen sistem kullanıcısıyla bir bağlantı kurduğunda ve kullanıcıyı sunucu ile aynı yükseltilmiş ayrıcalıklara sahip bir yöntem (child) ile doğrulamaya çalıştığında ortaya çıkar. Kötü niyetli bir kullanıcı bu noktada ayrıcalıklı yöntem (child) ile güvenlik açıklarından yararlanabilir ve

ayrıcalıklı sürecin kontrolünü veya erişimini kontrol altında tutabilir. Bu bağlamda da Ayrıcalık Ayrıştırma tasarım deseni Güvenilmez Ayrışma tasarım deseninin daha spesifik bir örneği haline gelmiş olur.

Bu tasarım deseni özellikle kullanıcıların kimliğini doğrulaması ve ardından kullanıcıların normal kullanıcı düzeyinde ayrıcalıklarla etkileşimli programlar çalıştırmalarına izin vermesi gereken sistem hizmetleri için özellikle yararlıdır.

Şekil 3.3'de ayrıcalık ayrıştırma (PrivSep) tasarım deseninin yapısını ve davranışını gösterir. Bu şema alt işlemler oluşturmak için UNIX altında (fork) işlevine başvuruda bulunmaktadır. Desen Windows işletim sistemlerinde, Süreci Oluştur () (CreateProcess) metodu ile alt işlem uygulamak için kullanılır.



Şekil 3.3. Ayrıcalık Ayrıştırma tasarım deseni yapısı

Ayrıcalık ayrıştırma tasarım deseninde ayrıcalıklı sunucu işlemi, sistem kullanıcısı, ayrıcalıklı müşteri süreci, kullanıcı imtiyazlı istemci süreci gibi süreçler yer alır. Ayrıcalıklı sunucu işlemi, alt süreçler tarafından ele alınacak olan işlevsellik için ilk isteklerin yerine getirilmesinden sorumludur.

Sistem kullanıcısı, sistemden bir miktar işlem yapmasını ister ve bu ilk işlevsellik isteği kullanıcı tarafından ayrıcalıklı sunucuya yönlendirilir. Bahsedilen kullanıcı elbette yerel veya uzak kullanıcı olabilir.

Ayrıcalıklı müşteri süreci, kullanıcının isteğinin doğrulanmasından sorumludur. Bunun güvenilir bir kullanıcıdan gelen geçerli bir istek olup olmadığı henüz bilinmediğinden alt süreçler işleme kimlik doğrulamasının ayrıcalıkları ile sınırlandırılır.

Kullanıcı imtiyazlı istemci süreci, sistem kullanıcısı ve istekleri onaylandıktan sonra ayrıcalıklı sunucudan uygun kullanıcı düzeyinde ayrıcalıklara sahip bir alt süreci oluşturur. Kullanıcı imtiyazlı alt süreç aslında sistem kullanıcısının istediğini yerine getirir.

Bu desen yükseltilmiş ayrıcalıklara sahip bir sürecin kontrolünü ele almaz. Böylece zararlı kimselerin neden olabileceği hasar sınırlandırılmış olur. Kod incelemeleri ve ek testler ile resmi doğrulama teknikleri gibi ek doğrulamalar izinsiz ayrıcalık artışını daha da azaltabilecek özel bir ayrıcalıkla yürütülen koda odaklanabilir.

Ayrıcalık ayrıştırma tasarım deseni, ön kimlik doğrulama ve ön kimlik doğrulama sonrası şeklinde iki aşamadan oluşmaktadır. Ön kimlik doğrulaması aşamasında kullanıcı bir sistem ile bağlantıya geçer ancak henüz kimliğini doğrulamaz. Dolayısıyla hiçbir erişim ve dosya sistemine işlem imtiyazı bulunmaz. Ön doğrulama sonrası kimlik doğrulama aşamasında ise kullanıcı sisteme başarıyla doğrulanır ve dosya sistemi erişimi gibi kullanıcı ayrıcalıklarına sahiptir.

Ayrıcalık ayrıştırma deseninde uygulanan genel işlem aşağıdaki gibidir:

- Ayrıcalıklı bir sunucu oluşturulur.
- İlk kullanıcı istekleri bu sunucuya yönlendirilir.
- Sunucuya bir kullanıcı isteği geldiğinde, sunucu kimlik doğrulama işlemi sırasında gereken kullanıcı etkileşimi için güvenilmeyen, ayrıcalıklı olmayan bir yöntem (child) ortaya çıkarır.
- Kullanıcının kimliği doğrulandıktan sonra, sunucu kullanıcının isteğini yerine getirmek için uygun bir Kullanıcı Kimliği (UserID) ile başka bir alt işlemden çıkar.

Linux altında fork (), chroot () ve stuid () yöntemleri kullanılarak ayrıcalık ayrıştırma tasarım desenine uygun uygulamalar yapılabilir.

Ayrıcalık ayrıştırma tasarım deseni ile ilgili bilinen uygulamalar aşağıdaki gibidir.

- OpenBSD (sshd, bgpd, ospfd, ripd, rtadvd, X sunucusu, snmpd, ntpd, dhclient, tcpdump)
- Güvenli XML-RPC Sunucu kütüphanesi

3.1.3 Çekirdeğe Erteleme Tasarım Deseni

Çekirdeğe erteleme (Defer to Kernel) tasarım deseninin amacı, yükseltilmiş ayrıcalıklar gerektiren işlevleri, yükseltilmiş ayrıcalıklar gerektirmeyen işlevsellikten açıkça ayırmak ve çekirdek düzeyinde mevcut olan kullanıcı doğrulama işlevinden yararlanmaktır. Mevcut kullanıcı doğrulama çekirdeği işlevselliğini kullanmak, kullanıcı düzeyindeki güvenlik kararlarını uygulama yollarını yeniden icat etmek yerine çekirdeğin güvenlik kararlarının uygulama etmedeki rolünü güçlendirmektedir. Çekirdeğe erteleme tasarım deseni ile referans izleme tasarım deseni arasındaki temel fark, çekirdeğe erteleme deseninin işletim sistemi çekirdeği tarafından sağlanan kullanıcı doğrulama işlevselliğinin kullanımına odaklanmasıdır. Oysa referans izleme tasarım deseni yetkilendirme yöntemini belirtmemektedir.

Bu güvenli tasarım deseni için birincil motivasyon, yükseltilmiş ayrıcalıklarla çalışan ve dolayısıyla ayrıcalık saldırılarına karşı duyarlı olan kullanıcı programlarına olan ihtiyacı azaltmak ve önlemektir. UNIX tabanlı sistemlerde bu durum kullanıcı kimliğinin ayarlandığı (setuid) programların azaltılması veya önlenmesi anlamına gelir. Windows tabanlı sistemlerde ise bu durum, yönetici olarak çalışan kullanıcı programlarından kaçınılması anlamında gelir. Ek olarak, bu tasarım deseni işletim sistemi çekirdeği tarafından sağlanan kullanıcı doğrulama işlevselliğinin yeniden kullanılmasına odaklanmaktadır. Kullanıcıları doğrulamak için mevcut çekirdek işlevselliğinin yeniden kullanılması aşağıdaki avantajları beraberinde getirmektedir.

- Geliştiricilerin kendi kullanıcı kimliklerini ve doğrulama işlevlerini yeniden yazmaları gerekmez,
- Mevcut çekirdek kullanıcı kimliği ve doğrulama işlevselliği üzerinde zaten test ve doğrulama yapıldığından bu işlemlerin tekrar yapılmasına gerek kalmaz,

- Daha fazla taşınabilir bir çözümdür çünkü her bir işletim sisteminin kullanıcıları her platformla tutarlı bir şekilde doğrulamasını sağlayacaktır.

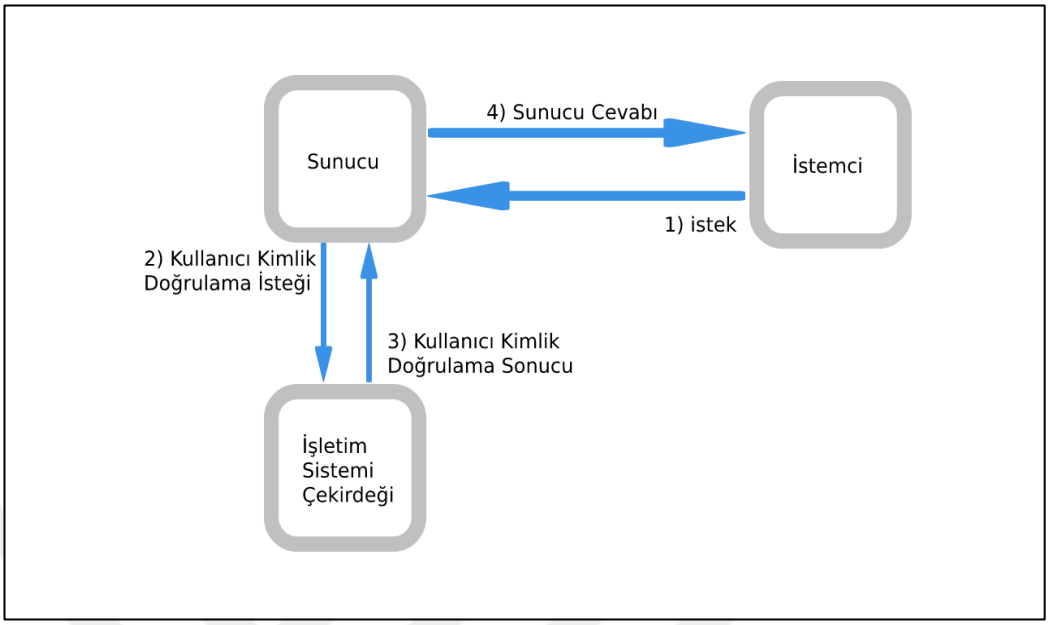
Eğer bir sistem aşağıdaki özelliklere sahip ise çekirdeğe erteleme tasarım deseni uygulanabilir.

- Sistem, ayrıcalıklara sahip olmayan kullanıcılar tarafından çalıştırılıyorsa,
- Sistemin işlevselliğinin bir kısmı yüksek ayrıcalıklar gerektiriyorsa,
- Yüksek ayrıcalıklar gerektiren işlevselliği yerine getirmeden önce, sistemin geçerli kullanıcının işlevini gerçekleştirmesine izin verildiğinin doğrulamasının gerektiği sistemlerde,

Özellikle UNIX tabanlı işletim sistemlerinde çalışan uygulamalar için, sistemin aşağıdaki özelliklere sahip olması durumunda da çekirdeğe erteleme tasarım deseni uygulanabilir.

- Program, görevlerinin bir kısmını veya tamamını yerine getirmek için özel bir Kullanıcı Kimliği (UID) altında çalışıyorsa,
- Program, kullanıcılar tarafından gönderilen dosyaları veya iş taleplerini kabul edebiliyorsa,
- Yerel kullanıcılar için, programın erişim kontrolünü veya muhasebe için her bir dosya veya iş talebini hangi Kullanıcı Kimliği (UID) veya Grup Kimliğine (GID) gönderdiğini biliyorsa,
- Yerel olmayan kullanıcılar için, program başka bir kullanıcı doğrulama ve kayıt mekanizması kullanıyor ise.

Çekirdeğe erteleme tasarım deseni temel bir istemci-sunucu mimarisini uygular. Sunucu yükseltilmiş ayrıcalıklarla çalışır, istemcilerden gelen kullanıcı isteklerini kabul eder ve mümkün olduğunda kullanıcıları doğrulamak için mevcut çekirdek işlevlerini kullanır. Çekirdeğe erteleme tasarım deseninin genel yapısı şekil 3.4'te gösterilmiştir.



Şekil 3.4. Çekirdeğe erteleme deseninin genel yapısı

Çekirdeğe erteleme tasarım deseninde Müşteri Programı, Sistem Çekirdeği ve Sunucu Programı bileşenleri gecikme düzenindeki katılımcıları belirtmektedir.

- Müşteri Programı (Client Program): istemci programı, standart kullanıcı seviyesi yetkileriyle çalışır. İstemcinin yeterli ayrıcalıklara sahip olmadığı işleri yapmak için sunucuya iş istekleri gönderir.
- İşletim Sistemi Çekirdeği (System Kernel): istemci ve sunucu arasındaki iletişim için kullanılan süreçler arası iletişim mekanizmasını yönetir, kullanıcı kimliği doğrulama ve erişim işlevselliğini sağlar. Çekirdeğe uygulanan önceden var olan işlevselliğin, sunucuda çalışmasına izin verilip verilmediğini kontrol etmek için kullanılır.
- Sunucu Programı (Server Program): sunucu programı IPC mekanizmasının tahsis edilmiş bir örneğini izler ve istemcilerden gelen iş taleplerini okur ve müşterilerinin gönderilen işlerin sunucuda çalıştırılıp çalıştırılmayacağını kontrol eder.

Daha önce tek bir çalıştırılabilir dosyaya dayanan uygulamalar, yeniden yapılandırılmalıdır. Bunlar UNIX tabanlı işletim sistemlerinde kullanıcı kimliği ayarlanabilen (setuid), Windows tabanlı işletim sistemlerinde ise yönetici olarak çalıştırılabilen dosyalardır.

Çekirdeğe erteleme tasarım deseninin bilinen uygulamaları aşağıdaki gibidir:

- Sunucu bilinen bazı mekanizmalar aracılığı ile müşteri isteklerini kabul etmeye başlar.
- İstemci sunucuya isteğe dahil olan müşteriyi tanımlayan bilgileri içeren bir istek gönderir. Bu bilgi kodlanmış veya mevcut bir kullanıcı kimliği kullanılarak gönderilmiş olabilir.
- Sunucu, kullanıcının isteğini alır ve kullanıcının isteğini karşılayıp karşılamadığını veya isteği reddetmek için bazı çekirdek düzeyi mekanizmaları kullanır.

Çekirdeğe erteleme tasarım deseni Windows'ta Güvenli Nesneler (Securable Objects in Windows) olarak da bilinmektedir [18].

3.1.4 Hatalarla Tam Görünüm Tasarım Deseni

Hatalarla tam görünüm (Full View With Errors) tasarım deseni, geliştiricilerin kullanıcı ara yüzü geliştirirken izinleri yok saymasına izin verir [19]. Bu desen, her kullanıcının, kullanıcının çağırmasına izin verilmeyen seçenekler de dahil olmak üzere bir programın seçeneklerini görmesini sağlar. Bir kullanıcı istenen seçeneği seçtiğinde, sistem kullanıcının bu işlemi kullanıp kullanamayacağını ve buna göre cevap verip vermeyeceğini belirler. Bu mimari desen, kullanıcıların yasa dışı işlem yapmasını önler ve neredeyse tüm ayrıcalıklara sahip kullanıcılar için çok faydalıdır.

Bu tasarım deseni aynı zamanda bir kayıt modülüne sahip olan hatalara cevap verebilecek bir hata yönetimi çerçevesiyle oluşturulmalıdır. Bu desenin kullanımı, eğitim materyallerinin tüm kullanıcılar için tutarlı olmasını sağlar. Bu desenin uygulanması kolaydır çünkü tek bir görünüm tüm kullanıcılara sunulabilir. Bununla birlikte, kullanıcıları hata mesajları ile bombardıman ederek karıştırabilir. Bu desenle

iyi çalışan diğer desenler arasında, işlemlerde güvenlik kontrolleri gerçekleştirme mekanizmaları ve bir kullanıcının istenen işlemi gerçekleştirme izni olup olmadığına karar vermek için roller bulunur [19].

3.2 Tasarım Düzeyi Güvenli Tasarım Desenleri

Tasarım düzeyi desenleri, üst düzey bir sistem bileşeninin elemanlarının nasıl tasarlanıp uygulandığını açıklar. Üst seviye bileşenlerin kendi tanımlarını ve etkileşimlerini değil, tek bir üst seviye bileşenin iç tasarımındaki sorunları ele alır. Tasarım düzeyi desenler:

- Güvenli Fabrika Tasarım Deseni
- Güvenli Strateji Fabrikası Tasarım Deseni
- Güvenli Oluşturucu Fabrikası Tasarım Deseni
- Güvenli Sorumluluk Zinciri Tasarım Deseni
- Güvenli Durum Makinesi Tasarım Deseni
- Güvenli Ziyaretçi Tasarım Deseni
- Kontrollü Nesne Fabrikası Tasarım Deseni

3.2.1 Güvenli Fabrika Tasarım Deseni

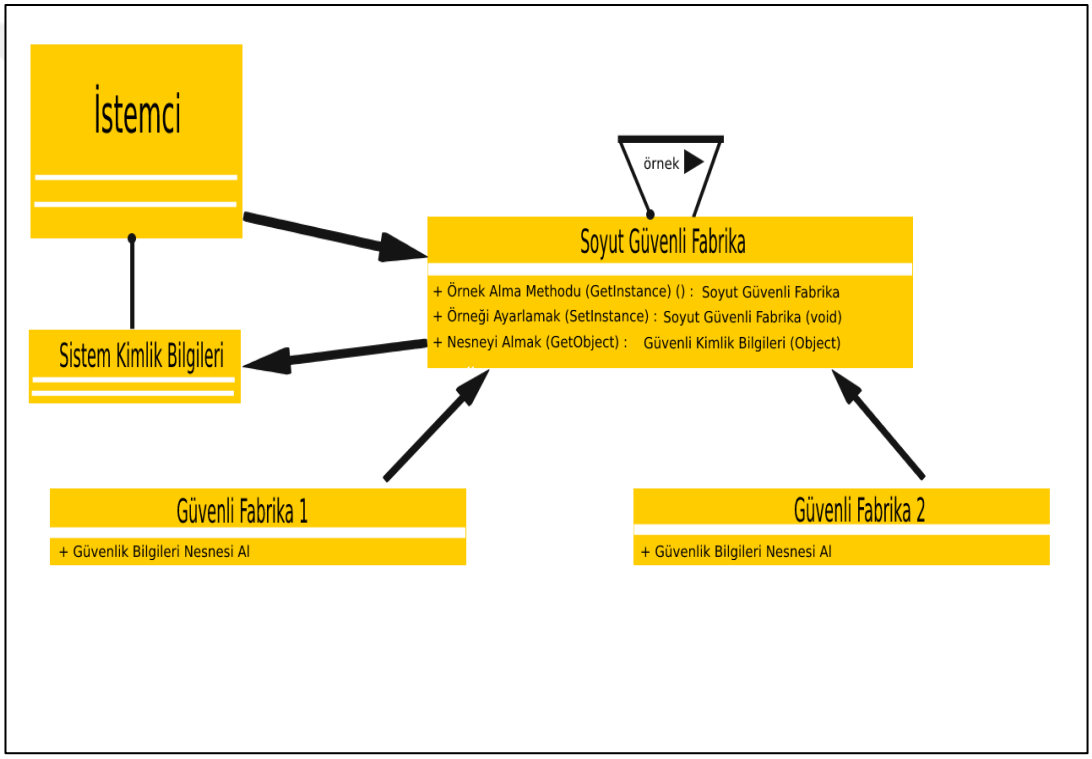
Güvenli Fabrika (Secure Factory) güvenli tasarım deseninin amacı, bir nesneyi oluşturma ya da seçmedeki güvenlik mantığını, oluşturulan ya da seçilen nesnenin temel işlemlerinden ayırmaktır. Güvenli fabrika tasarım deseni, soyut fabrika modelinin güvenliğe özgü bir uzantısıdır [20].

Kullanıcı güven seviyesine göre doğru nesneyi seçmek için gerekli olan mantık, güvenli fabrika tasarım deseni kullanılarak nesneyi ve nesneyi oluşturan uygulama kodundan ayrılabilir. Güvenlik tabanlı nesne seçimi mantığı ile nesne uygulaması arasındaki kararlı olmayan bağlantı ve güvenlik tabanlı nesne seçimi mantığı ile güvenli uygulama arasındaki kararlı olmayan bağlantı güvenlik tabanlı nesne seçimi mantığının doğrulanmasını, test edilmesini ve değiştirilmesini kolaylaştırır.

Güvenli fabrika tasarım deseni aşağıdaki durumlarda geçerli olmaktadır:

- Sistem, bir kullanıcının güvenlik kimlik bilgilerine dayanarak bir nesnenin farklı versiyonlarının oluşturulduğu senaryolarda,
- Mevcut güvenlik bilgilerinin, doğru nesneyi seçmek ve oluşturmak için gereken tüm bilgileri içerdiği durumlarda.

Şekil 3.5 güvenli fabrika tasarım modelinin yapısını göstermektedir.



Şekil 3.5. Güvenli Fabrika Tasarım Deseninin Yapısı

Çalışma zamanında güvenli fabrika örneğini değiştirme işlevi sunar ve güvenli fabrika sınıflarında uygulanacak bir nesneyi alma ara yüzünü tanımlar. Somut Güvenli Fabrika, Nesneyi Almak () yöntemini uygulayan soyut fabrikanın somut uygulamalarını temsil eder. İstemci sınıfı, kullanıcının güvenlik kimlik bilgilerini izler fabrikanın somut bir örneğini almak için Örnek Alma () yöntemini kullanır ve ardından somut fabrikanın Nesneyi Almak () özelliğini kullanarak verilen güvenlik bilgileri için bir nesne alır. Güvenlik Kimlik Bilgileri, geçerli kullanıcı kimlik bilgilerini temsil

eder. Güvenli fabrika modeli, güvenlik mantığını İstemciden ayırır ve gizler; böylece test edilmesi daha kolay olan daha kısa kod elde edilir. Güvenli Fabrika, Müşterinin davranışını değiştirmek zorunda kalmadan güvenlik mantığının değişmesini sağlayan kara kutu görevi görür.

- İstemci (Client): istemci, bir kullanıcının veya sistemin çalıştığı ortamın güvenlik bilgilerini izler. İlgili güvenlik kimlik bilgileri göz önüne alındığında, müşteri güvenli fabrikanın somut bir örneğini almak için Soyut Güvenli Fabrika sınıfının Örnek Alma () yöntemini kullanır ve ardından geçerli güvenlik verilen uygun nesneyi elde etmek için Nesneyi Alma () yöntemini çağırır.
- Sistem Kimlik Bilgileri sınıfı, bir kullanıcının güvenlik kimlik bilgilerini gösterir.
- Soyut Güvenli Fabrika sınıfı birkaç amaca hizmet etmektedir. Bunlardan bir tanesi fabrikanın Nesneyi alma () yöntemi ile güvenli bir fabrikanın somut bir örneğini sağlar. Diğeri sistemin, çalışma zamanında Örneği Ayarlama () yöntemi ile varsayılan somut güvenli fabrikayı ayarlamasını sağlar. Bu çalışma zamanında farklı bir somut güvenli fabrika belirleyerek nesne seçim metodolojisini değiştirmeyi nispeten kolaylaştırır.
- Soyut Güvenli Fabrika'nın tüm somut uygulamaları tarafından uygulanması gereken soyut Nesneyi Alma () yöntemini tanımlar. Nesneyi Alma () yöntemi, müşteri tarafından bazı güvenlik bilgileri verilen uygun nesneyi elde etmek için çağrılır.
- Somut Güvenli Fabrika N: Soyut Güvenli Fabrika'nın çeşitli somut uygulamalarında farklı nesne seçim metodolojileri uygulanmaktadır. Her bir somut güvenli fabrika, Nesneyi Alma () yönteminin bir uygulamasını sağlar.
- Çeşitli Nesne sınıfı, güvenli fabrika tarafından döndürülen nesneler tarafından uygulanan temel arabirimi tanımlar.
- Somut Nesne N: geçerli güvenlik bilgileri tarafından belirtilen her nesne davranışı kümesi için bir Çeşitli Nesne somut uygulaması oluşturur.

Güvenli fabrika, arayan kişiye uygun nesneyi sağlayan kara bir kutu olarak çalışır. Bu, güvenlik bağımlı nesne seçim mantığını arayandan gizler. Güvenli fabrika tarafından

oluşturulan nesnelerin yalnızca ilgili güven seviyelerine uygun işlevsellik uygulaması gerekir. Güvenli Fabrika tarafından oluşturulan nesnelerin, geçerli kullanıcı veya işletim ortamı hakkında bilgi verilen nesneye uygulanan bir eyleme izin verilip verilmediğini kontrol etmek gerekmez. Bu kontroller, nesneyi oluşturan Güvenli Fabrika tarafından zaten yapılmıştır.

Güvenli Fabrika tasarım deseninin kara kutu yapısı, sistemin güvenlik bilgilerine bağlı davranışını değiştirmeyi kolaylaştırır. Nesne seçim mantığındaki veya sağlanan nesnelerin kendisindeki değişiklikler, Güvenli Fabrikayı kullanan kodda çok az değişiklik yapılmasını gerektirecektir.

Güvenli fabrika tasarım deseninin uygulamadaki genel süreci şu şekildedir:

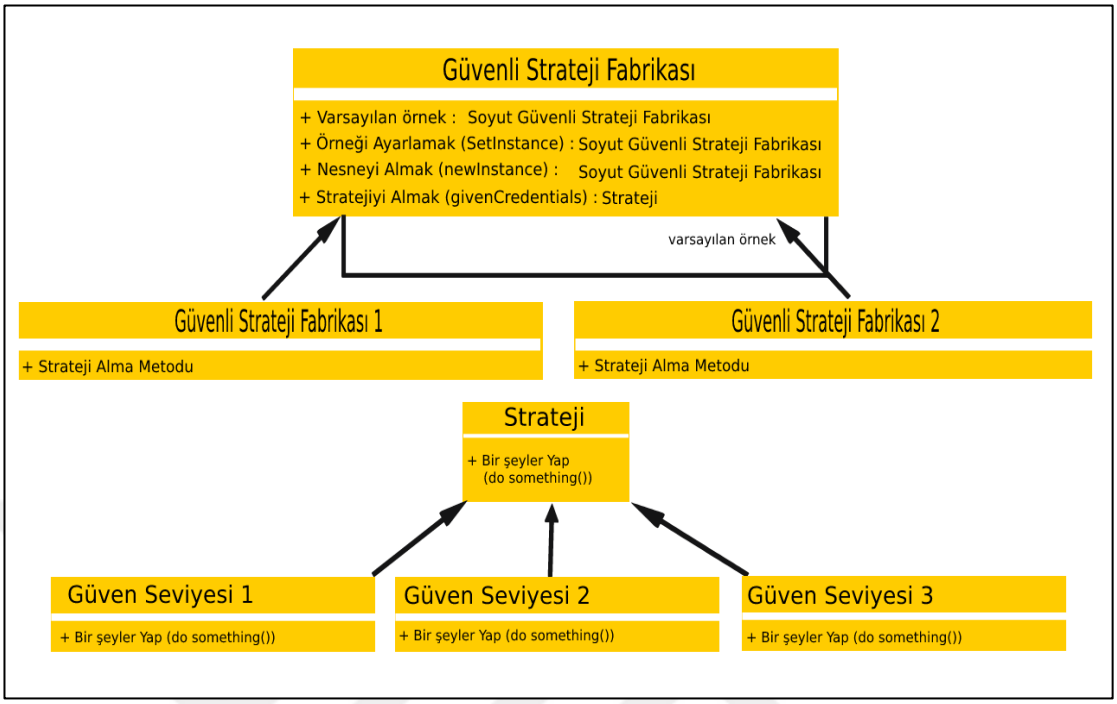
- Yapısı veya seçimi, kullanıcı veya işletim ortamıyla ilişkili güven düzeyine bağlı olan bir nesne tanımlanır.
- Nesne tarafından desteklenen genel işlevselliği tanımlayan soyut bir sınıf veya ara yüz tanımlanır.
- Nesnenin güven düzeyine özgü davranışını uygulayan somur sınıflar uygulanır. Tanımlanan her güven nesnesi için bir somut uygulama oluşturulur.
- Soyut Güvenli Fabrika sınıfını tanımlamak için temel soyut fabrika modeli kullanılır.
- Bir kullanıcının veya ortamın güven düzeyini belirlemek için gereken bilgileri belirlenir. Bu bilgi Güvenlik Bilgileri sınıfını veya veri yapısını tanımlamak için kullanılır.
- İkinci adımda tanımlanan nesne ile dördüncü adımda tanımlanan güvenlik kimlik bilgilerine göre seçen somut güvenli bir fabrika uygulanır.
- Beşinci adımda tanımlanan soyut fabrika tarafından sağlanan varsayılan fabrika olarak beşinci adımda tanımlanan somut güvenli fabrika ayarlanır.
- Yazılan uygulamanın, kullanılan güvenli fabrikayı kolayca değiştirmesini gerektirmiyorsa, soyut olmayan fabrika modeli kullanılarak güvenli fabrika tasarım desenini uygulamak mümkün olacaktır.
- Beşinci adımda tanımlanan soyut fabrika tarafından sağlanan varsayılan fabrika olarak beşinci adımda tanımlanan somut güvenli fabrika ayarlanır.

- Yazılan uygulamanın, kullanılan güvenli fabrikayı kolayca değiştirmesini gerektirmiyorsa, soyut olmayan fabrika modeli kullanılarak güvenli fabrika tasarım desenini uygulamak mümkün olacaktır.

3.2.2 Güvenli Strateji Fabrikası Tasarım Deseni

Güvenli strateji fabrikası (Secure Strategy Factory) tasarım deseni bir kullanıcının güvenlik kimlik bilgilerine göre gerçekleştirmek istediği bir işlem için uygun stratejide bir desen uygulayan nesnedir. Uygun desenin seçilmesinde güvenli strateji fabrikasının kullanımı kolaylık sağlamaktadır. Güvenli strateji fabrikası tasarım deseni, kullanıcının güvenlik bilgilerine dayanarak bir strateji nesnesi seçmenin kolay yolunu sunar. Bu tasarım deseni, sisteme giriş yapan kullanıcıya bağlı olarak değişiklik göstermesi gerektiğinde ve yalnızca kullanıcının kimlik bilgileriyle belirlenebildiği zaman kullanılmalıdır. Bu yüzden strateji adını da almıştır.

Güvenli olarak kabul edilen bir sistem içerisindeki fonksiyonlar, sistemdeki kullanıcıların veya sistemin çalıştığı ortamın güvenlik kimlik bilgilerine dayanarak farklı davranışlar gösterebilmektedir. Örnek vermek gerekirse desene uygun olarak çalışan bir sistemde, kullanıcının güven düzeyine bağlı olarak değişkenlik gösteren çeşitli hata mesajları yer alabilir. Güvenilmeyen kullanıcılar, güvenilen kullanıcılara göre daha fazla bilgi içermeyen hata mesajı alabilir. Güvenli strateji fabrikası deseni, genel bir sistem işlevi için güvenlikte kimlik bilgilerine dayanan bir strateji sergiler ve uygulanacak olan davranışların sistemden değiştirilmesi, test edilmesi ve doğrulanmasının daha kolay yapılması için davranışlar kodlama esnasında belirlenmelidir. Şekil 3.6. Güvenli Strateji Fabrikası desen yapısını göstermektedir.



Şekil 3.6. Güvenli Strateji Fabrika Desen Yapısı

Şekilde yer alan sınıflardan Güvenlik Bilgileri sınıfı, bir kullanıcının güvenlik bilgilerini vermektedir.

Soyut Güvenli Strateji Fabrikası sınıfı birkaç amaca hizmet etmektedir. Bunlardan bir tanesi fabrikanın Örneği Alma () yöntemi ile güvenli bir strateji fabrikasının somut örneğini sağlar diğeri ise sistemin çalışma esnasında Örneği Ayarlama () yöntemi ile gerçek somut güvenli stratejiyi ayarlamasına olanak sağlar. Soyut Güvenli Strateji Fabrikası sınıfının tüm somut uygulamaları tarafından uygulanması gereken soyut Stratejiyi Al () yöntemini tanımlamaktadır. Somut Güvenli Strateji Fabrikası N sınıfı Soyut Güvenli Strateji Fabrikası çeşitli somut uygulamalarında farklı strateji seçim yöntemleri uygulamaktadır. Strateji sınıfı, soyut strateji sınıfıdır. Genel bir sistem işlevini temsil eden bir yöntem tanımlamaktadır. Soyut strateji sınıfının tüm somut uygulamaları bu yöntemin uygulanmasında rol oynamaktadır.

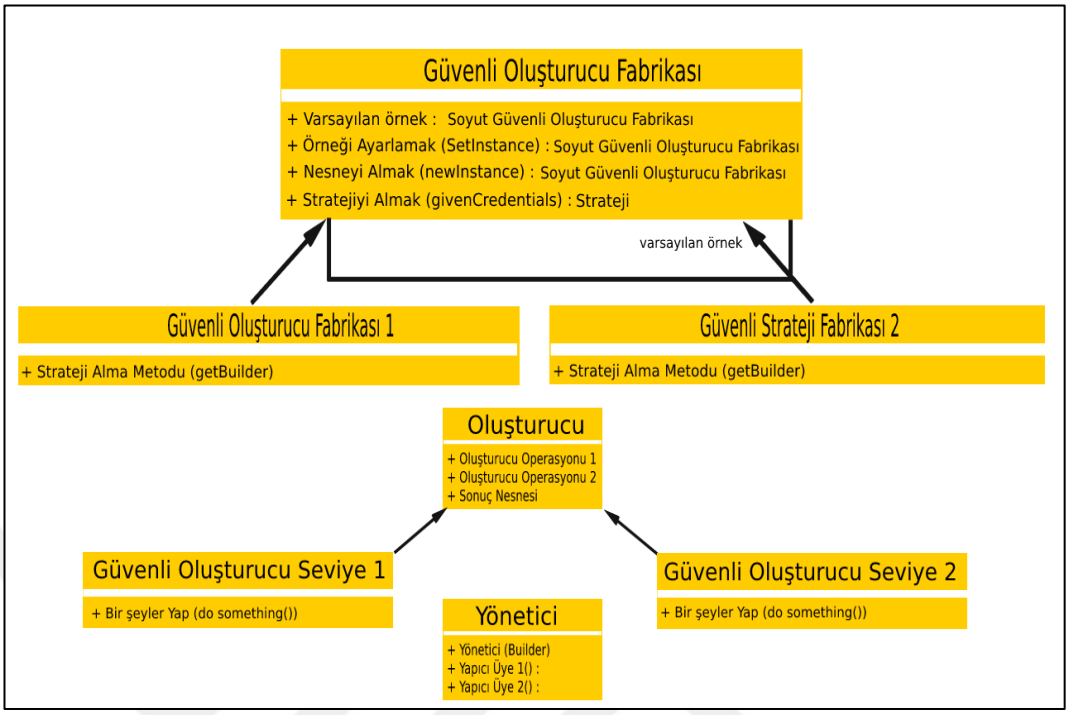
Güven Seviyesi Strateji N sınıfı, sistemdeki genel işlevlerin her biri için güvenlik kimlik bilgisi temelli bir yöntem olarak her güven düzeyi için bir somut Güven Seviyesi Strateji uygulanmasını sağlamaktadır.

Güvenli strateji fabrikası deseninin uygulamadaki genel süreci aşağıdaki gibidir.

- Güvenli tasarım desenine uygun genel bir sistem işlevi tanımlanır.
- Tanımlanan genel sistem fonksiyonunu uygulayan sınıfların ara yüzünü tanımlamak için strateji deseni kullanılır.
- İkinci adımda tanımlanan arabirimin somut uygulamaları yazılır. Bunu nesne tabanlı programlamadaki soyut sınıflardaki metotlarla iletişim kuran somut sınıflara benzetebiliriz. Bu yüzden genel sistem fonksiyonunun her bir güvenlik sürümü için somut bir uygulama yazılmalıdır.
- Soyut Güvenli Strateji Fabrikası sınıfını uygulamak için temel soyut fabrika modeli kullanılır.
- Kullanıcının güven düzeyini belirlemek için gereken bilgiler belirlenir ve bu bilgiler Güvenlik Bilgileri sınıfını veya veri yapısını tanımlamak için kullanılmaktadır.
- Dördüncü adımdaki güvenlik belgesi verildiğinde üçüncü adımda tanımlanan uygun stratejiyi seçen somut bir güvenli strateji fabrikası kurulur.
- Son olarak beşinci adımda tanımlanan somut güvenli strateji fabrikası, üçüncü adımda tanımlanan soyut fabrika tarafından sağlanan varsayılan fabrika olarak ayarlanır.

3.2.3 Güvenli Oluşturucu Fabrikası Tasarım Deseni

Güvenli oluşturucu fabrikası (Secure Builder Factory) tasarım deseni, güvenlik kimlik bilgilerine göre bir strateji seçmenin kolay bir yolunu sunar. Bu model güvenli fabrika tasarım deseninden genişletilmiştir. Güvenli oluşturucu fabrikası, sisteme giriş yapan kullanıcıya bağlı olarak değişiklik göstermesi gerektiğinde ve yalnızca kullanıcının kimlik bilgileriyle belirlenebildiği zaman kullanılmalıdır. Bu desenin amacı, karmaşık bir nesne oluşturulurken güvenliğe bağlı kuralları, nesneyi gerçekten oluşturmaya dahil olan temel adımlardan ayırmaktır. Güvenli oluşturucu fabrikası tasarım deseni güvenlik kuralları uyarınca karmaşık bir nesne oluşturacak olan oluşturucu desenini uygulayan uygun nesneyi seçmek ve iade etmek için verilen güvenlik bilgilerini kullanmaktadır. Şekil 3.7 güvenli oluşturucu fabrikası tasarım deseninin yapısı gösterilmektedir.



Şekil 3.7. Güvenli Oluşturucu Fabrika Tasarım Deseni Yapısı

Şekil 3.7'de yer alan sınıflardan Yönetici sınıfı, oluşturucu deseninin yönetmen öğesidir. Yönetmen oluşturucu örneği karmaşık nesneler oluşturur. İstemci, güvenli oluşturucu fabrikası tarafından seçilen bir oluşturucu ile yönetmeni harekete geçirir ve daha sonra belirli karmaşık nesneler oluşturmak için Yönetici içerisindeki yöntemleri çağırır. Güvenlik Bilgileri sınıfı, bir kullanıcının güvenlik kimlik bilgilerini göstermektedir. Soyut Güvenli Oluşturucu Fabrikası sınıfı, birkaç amaca hizmet eder. Bunlardan bir tanesi fabrikanın Örneği Alma () yöntemi ile güvenli bir oluşturucu fabrikasının örneğini sağlar. Diğeri ise sistemin güvenli oluşturucu fabrikası çalışma zamanında Örneği Ayarlama () yöntemi ile ayarlanmasını sağlar. Bu çalışma süresinde oluşturucu seçim metodolojisini değiştirmeyi kolaylaştırmak amacıyla farklı bir somut güvenli oluşturucu fabrikası belirler. Soyut Güvenli Oluşturucu Fabrikası sınıfının tüm somut uygulamaları tarafından uygulanması gereken soyut Oluşturucuyu Alma () yöntemini tanımlar. Somut Güvenli Oluşturucu Fabrikası N, Soyut Güvenli Oluşturucu Fabrikası sınıfının çeşitli somut uygulamalarında farklı oluşturucu seçim yöntemleri uygulamaktadır. Oluşturucu, soyut oluşturucu sınıfı karmaşık nesnenin tüm

oluşturucuları için geçerli olan temel oluşturucu yöntemlerini tanımlar. Güvenli Oluşturucu Seviyesi N, soyut oluşturucu sınıfının bir somut uygulamasıdır.

Güvenli Oluşturucu Fabrikası tasarım deseninin uygulanmasındaki işlemler aşağıdaki gibidir.

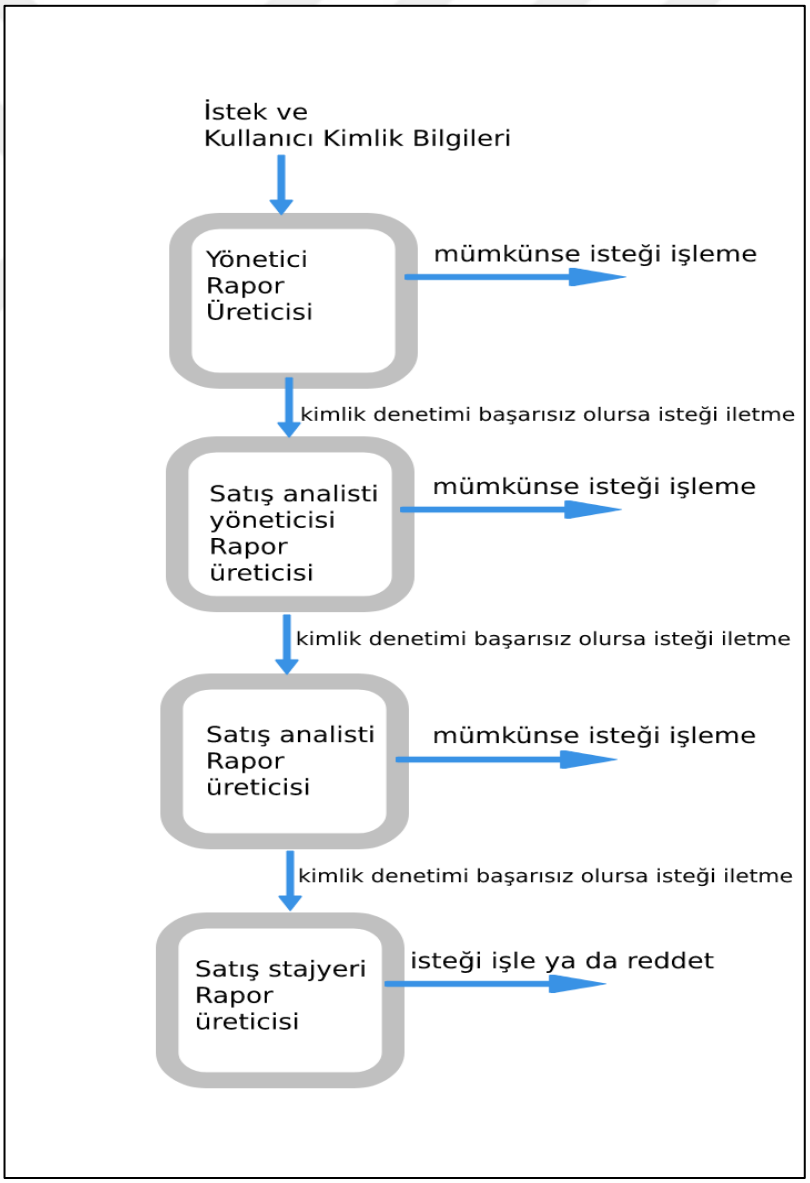
- Yapısı kullanıcı veya işletim sistemi ile ilişkili güven düzeyine bağlı olan karmaşık bir nesne tanımlanır.
- Bu tür nesneleri tanımlamak için oluşturucu deseninin kullanarak oluşturucu arabirimi tanımlanabilir.
- Karmaşık nesne için çeşitli güven düzeyine özgü yapı kurallarını uygulayan somut oluşturucu sınıfları uygulanır.
- Kullanıcıların güven düzeyini belirlemek için gereken bilgiler belirlenmelidir. Bu bilgiler Güvenlik Bilgileri sınıfını veya veri yapısını tanımlamak üzere kullanılacaktır.
- Güvenlik bilgileri verilen ve uygun oluşturucu seçen bir güvenli oluşturucu fabrikası uygulanır.

3.2.4 Güvenli Sorumluluk Zinciri Tasarım Deseni

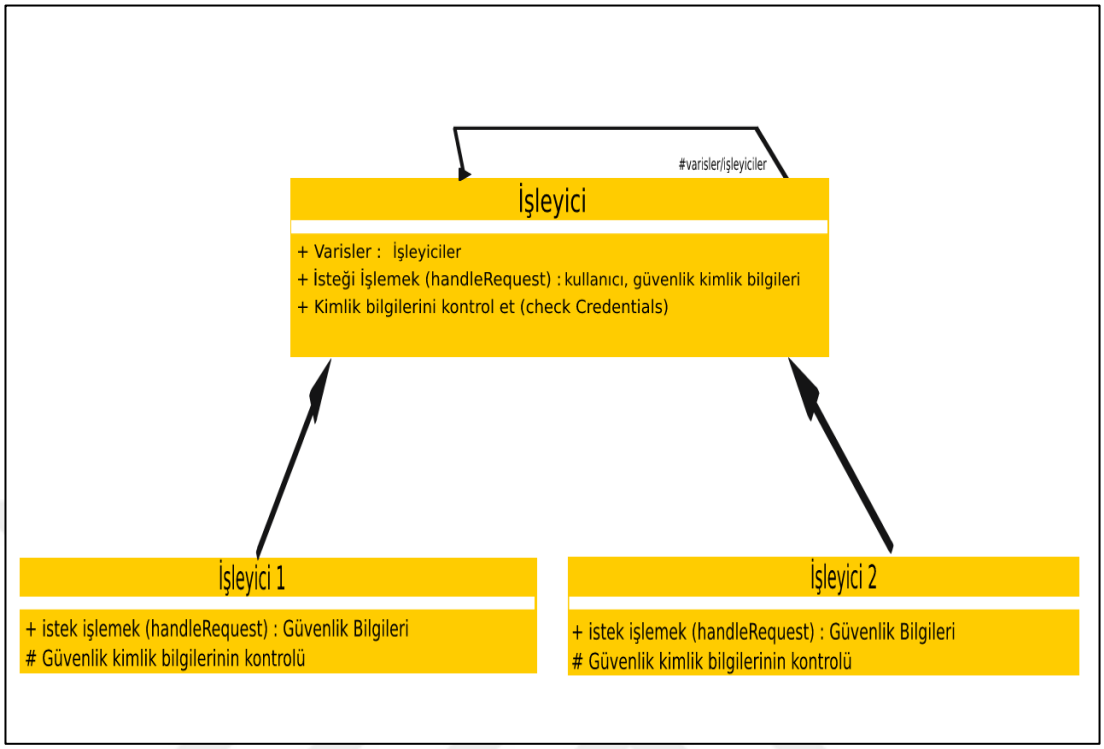
Güvenli sorumluluk zinciri (Secure Chain of Responsibility) tasarım deseninin amacı, kullanıcı bağımlı işlemlerin gerçekleştirildiği senaryolarda, kullanıcıya bağımlı işlemleri belirleyen mantığı basitleştirmektir. Rol tabanlı erişim ve kontrol mekanizması kullanan bir uygulamada çeşitli sistem fonksiyonlarının davranışı mevcut kullanıcının rolüne bağlı olarak gerçekleşir.

Genel bir sistem işlevi için role dayalı davranışlar, kullanıcıya genel bir sistem işlevini gerçekleştirebilmesi ile ilgili olarak düşük veya yüksek seviyede rolüne dayanarak erişim izni vermek veya vermemek şeklinde gerçekleşir. Genel bir sistem işlevi için uygun olan davranışın rol temelli seçimini tek bir kritere bağlı uygulamak yerine, güvenli sorumluluk zinciri tasarım deseni ile doğru davranışı seçme mantığını kullanarak işletilmesi sağlanır.

Bir rapor oluşturma modülü düşünüldüğünde kullanıcıların raporları yalnızca kendilerine tanımlanmış olan rollere göre oluşturabildikleri söylenebilir. Örneğin genel yönetici rolüne sahip bir kullanıcı mevcut tüm bilgileri içeren raporları oluşturabilir. Ancak satış birimindeki yönetici yalnızca satış verilerini içeren konularda tüm raporları oluşturabiliyorken satış uzamanı rolündeki kullanıcı yalnızca satış verilerinin bir alt kümesini içeren raporlar oluşturabilecektir. Bu roller göz önüne alındığında Güvenli Sorumluluk Zinciri güvenli tasarım deseni, bu örnekteki gibi rapor oluşturma işlevselliğini uygulamak için şekil 3.8'deki gibi kurgulanmalıdır. Güvenli sorumluluk zinciri tasarım deseninin yapısı ise şekil 3.9'da gösterilmiştir.



Şekil 3.8. Güvenli Sorumluluk Zinciri Tasarım Deseni Rapor Oluşturma Örneği



Şekil 3.9. Güvenli Sorumluluk Zinciri Tasarım Deseni Yapısı

İşleyici sınıfların İsteği İşle () yöntemi, kullanıcının güvenlik kimlik bilgilerini argüman olarak alır. Sistemdeki kimlik bilgilerinin güvenlik hususunda sınıfın bir isteği işleme izni olup olmadığını belirlemek için bir Kimlik Bilgilerini Kontrol Et () yönteminin uygulanması gerekir. Eğer yapılan güvenlik kontrolü önemsiz ise Kimlik Bilgilerini Kontrol Et () yöntemi uygulanmayabilir. Güvenlik Bilgileri sınıfı bir kullanıcının güvenlik kimlik bilgilerini gösterir. Somut İşleyici N, İşleyici sınıfının somut bir uygulaması olarak İsteği İşlemek () yöntemini uygular.

3.2.5 Güvenli Durum Makinesi Tasarım Deseni

Güvenli durum makinesi (Secure State Machine) tasarım deseninin amacı, güvenlik ve kullanıcı düzeyinde işlevselliği uygulayarak güvenlik mekanizmaları ve kullanıcı düzeyinde işlevsellik arasında net bir ayrım sağlar.

Güvenli bir sistemin çalıştırılmasında iç içe geçmiş güvenlik ile ilgili tüm işlevsellik ve kullanıcı düzeyindeki diğer işlevsellikler karmaşıklığı arttırabilmektedir. Artan karmaşıklık uygulamanın güvenlik özelliklerini test etmeyi, incelemeyi ve doğrulamayı zorlaştırarak buradan da bir güvenlik açığı verilmesi olasılığını arttırır. Ayrıca güvenlik işlevi ile kullanıcı düzeyindeki işlevsellik arasındaki sıkı bağlantı, sistemin güvenlik mekanizmalarının değiştirilmesini zorlaştırmaktadır. Desen aynı zamanda güvenlik ve kullanıcı düzeyinde işlevsellik arasındaki eşleşmeyi azaltarak sisteme gelecekteki değişikliklerin kolayca uygulanabilmesini sağlar. Şekil 3.10'da güvenli durum makinesi tasarım desenine ait yapı gösterilmiştir.

Şekil 3.10. Güvenli Durum Makinesi Tasarım Deseni Yapısı

Güvenlik İçeriği, kullanıcıların ilgilendiği ara yüzü tanımlar. Kullanıcı Fonksiyon İçeriği sınıfını korur. Kullanıcı Fonksiyon İçeriği sınıfı için Proxy işlevi görür. Güvenlik Durumu, güvenlik durumu makinesi tarafından yönetilen olası işlemleri temsil eden ara yüzü tanımlar.

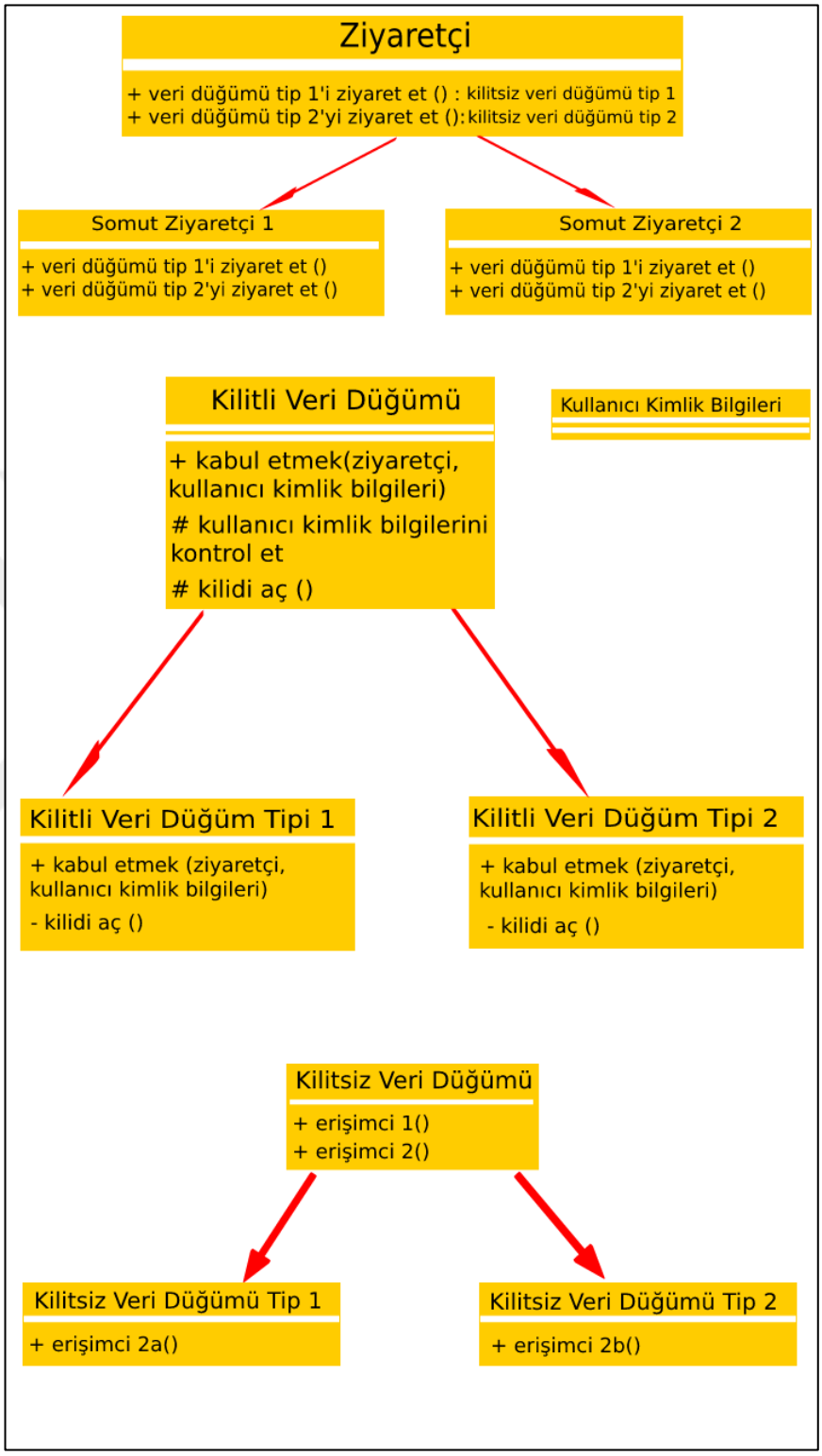
Güvenlik Somut Durumu sınıfında her Güvenlik Durumu alt sınıfı her işlem için güvenlik durumuna bağlı davranışı uygular. Kullanıcı Fonksiyon İçeriği, güvenlik durumu makinesinin bileşenlerinin uygun olduğunda işlem isteklerini kullanıcı düzeyinde durum makinesine iletebilmesi için Güvenlik İçeriği ile aynı işlemleri tanımlar. Kullanıcı Fonksiyon Durumu, Güvenlik Durumu ile aynı ara yüze sahiptir.

Kullanıcı Fonksiyonu Somut Durumu, Güvenlik Somut Durumuna benzer bir şekilde güvenlik durumunun her bir alt sınıfı her işlem için kullanıcı düzeyinde duruma bağlı davranışını uygular.

3.2.6 Güvenli Ziyaretçi Tasarım Deseni

Güvenli Ziyaretçi (Secure Visitor) tasarım deseni, kullanıcı uygun yetkilerden yoksun olduğunda düğüme erişimi engelleme araçları sağlar. Güvenli sistemler, veri hiyerarşisindeki her bir düğümün farklı erişim kısıtlamalarına sahip olabileceği, sistemlerdir. Farklı düğümlerdeki verilere erişim, verilere erişen kullanıcının rolüne bağlı olabilir. Güvenli ziyaretçi tasarım deseninde ziyaretçi, düğümün kilidini açmak için uygun kimlik bilgilerini sağlamadıkça düğümlerin bir ziyaretçi tarafından okunmaya karşı kendilerini kilitlemelerine olanak tanır. Güvenli ziyaretçi tasarım deseni, kilitli bir düğüme erişmenin tek yolunun bir ziyaretçiyle olduğu ve veri yapısındaki düğümlere yetkisiz erişimi önlemeye yardımcı olacak şekilde tanımlanır.

Güvenli durum makinesi tasarım deseninde olduğu gibi, güvenli ziyaretçi tasarım deseninin temel motivasyonu, güvenlikle ilgili konular ve kullanıcı düzeyinde işlevsellik arasında temiz bir ayrım sağlamaktır. Güvenli ziyaretçi tasarım deseni, tüm güvenlik hususlarını veri hiyerarşisindeki düğümlere tahsis eder ve geliştiricilerin yalnızca kullanıcı düzeyinde işlevsellik ile ilgilenen ziyaretçiler yazmasına izin vermez. Düğümleri veri hiyerarşisindeki güvenlik işlevlerinden yalnızca sorumlu kılmak, güvenlik işlevini kullanıcı düzeyindeki işlevsellikten daha titiz bir şekilde test etmeyi ve doğrulamayı daha uygun hale getirmeyi sağlar. Şekil 3.11'de güvenli ziyaretçi tasarım deseninin yapısı gösterilmiştir.



Şekil 3.11. Güvenli Ziyaretçi Tasarım Deseni Yapısı

Ziyaretçi: Güvenli ziyaretçi modelindeki ziyaretçi katılımcısı standart ziyaretçi şablonundaki ziyaretçi katılımcısıyla hemen hemen aynıdır. Modellerdeki birincil fark, çeşitli ziyaret yöntemlerinin Güvenli Ziyaretçi deseninde kilitsiz düğüm nesnelerini almasıdır. Oysa standart Ziyaretçi deseninki ziyaret yöntemleri basitçe bir düğüm nesnesi alır. Yani standart desende kilitli ve kilitsiz veri düğümleri kavramı yoktur.

Somut Ziyaretçi: standart Ziyaretçi deseninde olduğu gibi Somut Ziyaretçi sınıfları, soyut ziyaretçi sınıfında tanımlanan işlemleri uygular.

Kilitli Veri Düğümü sınıfı, bir ziyaretçiyi kabul eden bir Kabul () işlemini tanımlar. Ek olarak, Kilitli Veri Düğümü sınıfı bir kullanıcının kimlik bilgilerini kontrol etmek ve geçerli kilitli düğümün kilidini açmak için bir işlem tanımlar. Kilitli bir düğümün, düğümdeki verileri görüntülemek veya düğümdeki verileri değiştirmek için ortak işlem yapmadığı unutulmamalıdır. Düğüme olan tüm erişim, düğümün Kabul () işlemi aracılığı ile yönlendirilmelidir.

Kabul () işlemi: kullanıcının kimlik bilgilerini kontrol eder. Kimlik bilgileri, kullanıcının geçerli düğümdeki verileri görüntülemesi için geçerliyse, düğüm Kilidi Aç () işlemini kullanarak kendi kilidini açar ve kilidini açmış sürümünü ziyaretçinin ziyaret yöntemine geçirir.

Kilitli Veri Düğümü Tipi N sınıfları: soyut Kilitli Veri Düğümü sınıfında tanımlanan işlemleri uygular. Bu çeşitli kilitli düğüm nesnelerinin kilidini açmak ve düğümlerin kilitsiz sürümlerini döndürmek için Kilidi Aç () işlemini içerir.

Kilitsiz Veri Düğümü sınıfı: kilitli bir veri düğümünün kilitsiz sürümünü temsil eder.

Kilitsiz Veri Düğümü Tipi N: Kilitsiz Veri Düğümü öğesinin somut uygulamaları soyut sınıfta tanımlanan işlemleri uygular.

Kullanıcı Kimlik Bilgileri sistemin mevcut kullanıcısını veya mevcut kullanıcıya atanan izinleri temsil eder. Güvenli Ziyaretçi tasarım deseni, kullanıcı kimlik bilgilerinin uygulamasına çok fazla kısıtlama getirmez. Tek gereksinim, bir düğümün verilerine erişimi denetlemek için kimlik bilgilerini kullanmasının mümkün olmasıdır.

Güvenli ziyaretçi tasarım deseninin amacı, düğüm için uygun kimlik bilgilerini sağlamadan, verileri kilitli bir düğümde okumayı zorlaştırmaktır. Desenin kendisi uygun kimlik bilgileri olmadan kilitli bir düğüm verisini programatik olarak okumayı zorlaştırsa da kilitli düğümü oluşturan ham baytları okumak ve böylece kilitli düğümdeki verilere erişim sağlamak hala mümkün olabilir. Bu, kilitli düğümdeki verilerin aslında bir şekilde "kilitli" olması gerektiği anlamına gelir. Veri şifreleme veya çevrimdışı depolama kullanılarak kilitli bir düğümde kilitlenebilir.

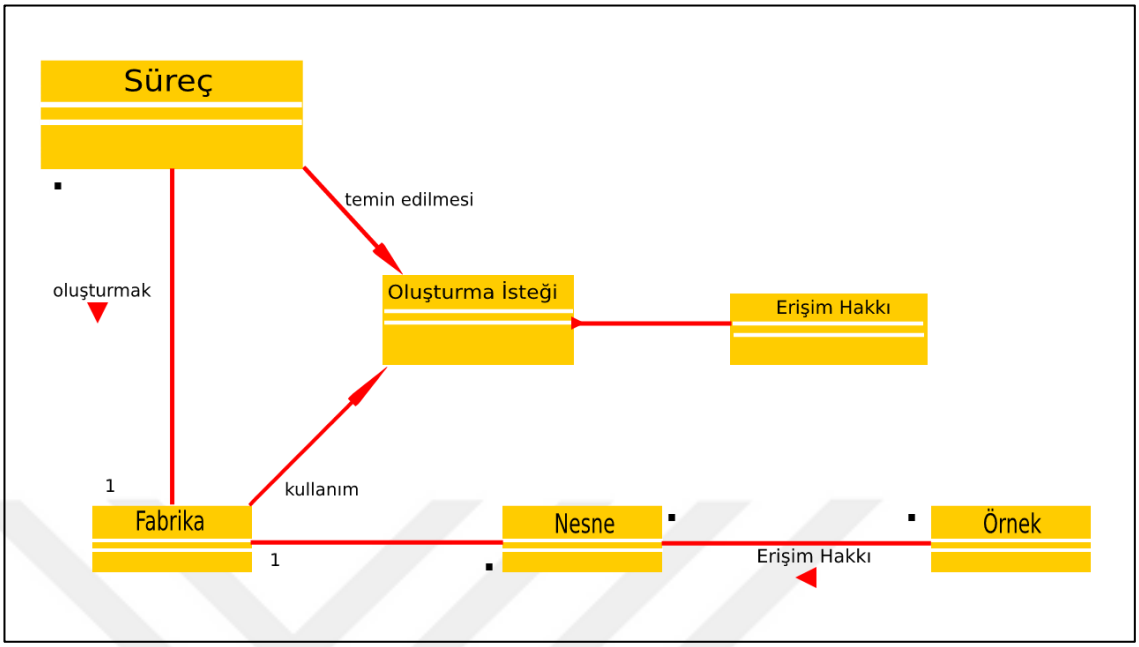
Şifreleme (Encryption): kilitli bir düğümdeki veriler şifrelenebilir ve yalnızca bir ziyaretçinin kimlik bilgileri kabul ettikten sonra düğümün kilitlenmemiş bir sürümünü yapma işleminin bir parçası olarak deşifre edilebilir.

Çevrimdışı Depolama: kilitli bir düğümdeki gerçek veriler, bir veri tabanı gibi bir tür harici, korumalı veri yönetim sisteminde saklanabilir. Gerçek düğüm verileri yalnızca bir ziyaretçinin kimlik bilgilerini kabul ettikten sonra harici kaynaktan yüklenir.

3.2.7 Kontrollü Nesne Fabrikası Tasarım Deseni

Kontrollü nesne fabrikası (Controlled Object Factory) tasarım deseni, nesne oluşturma sırasında izinleri belirleyip ayarlayarak nesneler için işlem izinlerinin yönetilmesine yardımcı olur. Her bir nesne için izinler değişmez ise, en az ayrıcalık ilkesi uygulanamaz. Dinamik izinlerin eksikliği, nesneleri kötüye kullanımlarına karşı savunmasız bırakabilir. Bu desen, nesneler oluşturulduğunda bir konu listesi ve ayrıcalıklar tanımlar.

Güvenlik politikaları, nesnelere kimlerin erişebileceğini açıklarken bu desen kullanılmalıdır. Bu deseni uygulamak için bir yöntem erişim kontrol listelerinin (EKL-ACL) nesnelerle ilişkilendirilmesini içerir. Sistemlerin kullanılabilir olması için nesnelerin izinlerinde dinamik değişikliklere izin vermeleri gerekir. Bu deseni kullanırken, varsayılan izinler olmayacağından, tüm nesnelerin tanımlanmış izinleri olmalıdır. Kaynak havuzlarından alınan nesneler dinamik olarak ayarlanmış izinlere sahiptir.



Şekil 3.12. Kontrollü Nesne Fabrikası Tasarım Deseni Yapısı

Şekil 3.12 altı katılımcıyla Kontrollü Nesne Fabrikasını göstermektedir. Süreç, Fabrika, Oluşturma İsteği, Nesne, Konu ve Erişim Hakkı şekilde yer alan katılımcıları göstermektedir.

İşlem: Nesnenin yaratılmasını ister.

Fabrika: Yaratma İsteğini kullanarak Nesneyi yaratır. Yaratma İsteği, İşlem tarafından gönderilir ve Nesne ile Konu arasında Erişim Hakkı içerir.

Konu: Nesneyi kullanmak isteyebilecek diğer nesneleri temsil eder. Kontrollü Nesne Fabrikası, nesnelerin onu kullanacak tüm konular için belirlenmiş uygun yetkilere sahip olmasını sağlar.

3.3 Uygulama Düzeyi Güvenli Tasarım Desenleri

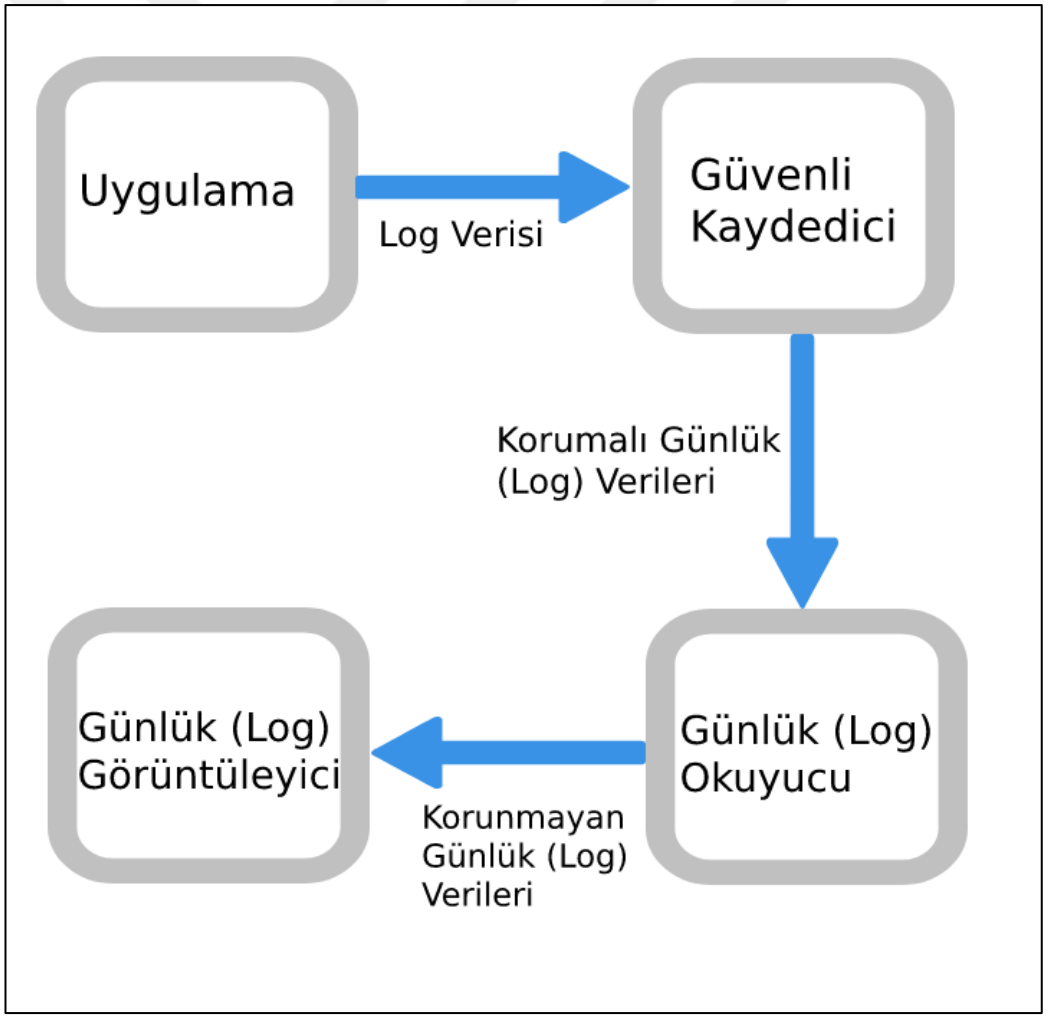
Uygulama düzeyindeki tasarım desenleri düşük seviyeli güvenlik sorunlarına yöneliktir. Bu sınıftaki desenler genellikle sistemdeki belirli işlevlerin veya yöntemlerin uygulanması için geçerlidir. Uygulama seviyesindeki modeller, CERT güvenli kodlama standartları tarafından belirlenen aynı problemi ele alır ve sıklıkla ilgili güvenli kodlama kılavuzuna bağlanır [21]. Bu başlık altında incelenecek olan uygulama düzeyi tasarım desenleri aşağıdaki gibidir.

- Güvenli Kaydedici Tasarım Deseni
- Hassas Bilgileri Temizle Tasarım Deseni
- Güvenli Dizin Tasarım Deseni
- Yol Adı Kanonikleştirme Tasarım Deseni
- Kaynak Toplama Başlatma Tasarım Deseni
- Giriş Doğrulama Tasarım Deseni
- Kimlik Doğrulayıcı Tasarım Deseni
- Yetkilendirme Tasarım Deseni
- Kontrol Noktası Tasarım Deseni
- Bilgi Gizliliği Tasarım Deseni
- Sınırlı Görünüm Tasarım Deseni
- Çok Düzeyli Güvenlik Tasarım Deseni
- Rol Tabanlı Erişim Kontrolü Tasarım Deseni
- Rol Hakları Tanımı Tasarım Deseni
- Roller Tasarım Deseni
- Tek Erişim Noktası Tasarım Deseni
- Güvenli Oturum Tasarım Deseni
- Güvenli Erişim Katmanı Tasarım Deseni
- Güvenli Kanallar Tasarım Deseni

3.3.1 Güvenli Kaydedici Tasarım Deseni

Güvenli kaydedici (Secure Logger) tasarım deseninin amacı, bir saldırganın sistem hakkında potansiyel olarak yararlı bilgiler toplayabilmesini ve sistem günlüğünü düzenleyerek saldırganın eylemlerini gizlemesini önlemektir.

Sistem günlükleri genellikle bir sistemi yöneten veya yeni bir sistemde hata ayıklayan kişiler için yararlı olan çok fazla bilgi içerir. Ayrıca bir sistem günlüğünde yer alan bilgiler, sisteme saldırmak için yeni vektörler tasarlamak için oldukça elverişli bilgileri içerebilir. Şekil 3.13 güvenli kaydedici fabrikası tasarım deseninin yapısını göstermektedir.



Şekil 3.13. Güvenli Kaydedici Fabrikası Tasarım Deseni Yapısı

Uygulama: güvenli kayıt sistemi tarafından işlenen ve depolanan kayıt verilerinin oluşturan bir tür ana uygulamadır.

Güvenli Kayıt Cihazı: güvenli kayıt sistemi kayıt bilgilerini yetkisiz bir kullanıcının kayıt verilerine erişmesini zorlaştıracak şekilde saklamaktan sorumludur.

Günlük Okuyucu güvenli günlük kaydı sistemi: günlük verilerini korumalı bir şekilde sakladığından, günlük kayıtlarının içeriğini okumak için güvenli günlük kaydı sistemine özgü bir okuma mekanizmasının kullanılması gerekecektir. Günlük dosyalarını okumak için standart mekanizmalar, örneğin dosyayı standart bir metin editöründe okumak veya kelime işlem uygulaması, güvenli kayıt sistemi tarafından depolanan kayıt verileri için çalışmayacaktır.

Günlük Görüntüleyici: yetkili bir kullanıcı bir tür günlük görüntüleme uygulamasını kullanarak günlük verilerini okuyabilir. Sonuç olarak güvenli günlük alt sistemi tarafından yönetilen günlük verilerine erişen saldırgan, gerçek günlük verileri içeriğini sınırlı olarak görecek veya hiç göremeyecektir. Bu sayede saldırgan, sistemde daha karmaşık saldırıları planlamak için günlük verilerindeki bilgileri kullanamayacaktır. Ayrıca bir saldırgan tarafından günlük verilerinde yapılacak olan saldırı amaçlı değişiklikler yetkili kullanıcı tarafından tespit edilebilir.

Bilinen güvenli günlük kayıt sistemleri aşağıdaki gibidir.

- **syslog-ng** (http://www.balabit.com/network-security/syslog)
- Syslog günlük kaydı uygulaması, bir Linux sunucusunda çalışan birden fazla uygulama tarafından kullanılabilecek güvenli bir merkezi kayıt sistemi sağlar.
- **smartInspect** (http://www.gurock.com/smartinspect)
- SmartInspect günlük kitaplığı, Java, Delphi ve .NET için AES şifreli günlük dosyalarını destekler.
- **CLoglt** (http://www.codeproject.com/KB/files/logit.aspx?msg=686567)

CLoglt uygulaması, şifreli günlük dosyalarının okunmasını ve yazılmasını sağlayan, kaynak kodlu küçük bir kütüphanedir. Windows için C++ programlama dili ile yazılmıştır.

Güvenli günlük kaydını desteklemek için genel dosya sistemi şifreleme yardımcı programlarını kullanmak da mümkündür. Bu durumda, günlük dosyaları şifrelenmiş bir dosyaya veya disk birimine yazılır ve okunur. Bunlar aşağıdaki gibidir.

- **Windows XP Encrypting File System (EFS)** (http://support.microsoft.com/kb/307877)
- Windows XP şifreleme dosya sistemi doğrudan Windows XP işletim sistemi tarafından desteklenen bir dosya veya klasör düzeyinde şifreleme özelliğidir.
- **TrueCrypt** (http://www.truecrypt.org)
- TrueCrypt, çeşitli işletim sistemleri için üçüncü taraf bir disk şifreleme aracıdır.
- **EncFS** (http://www.arg0.net/encfs)

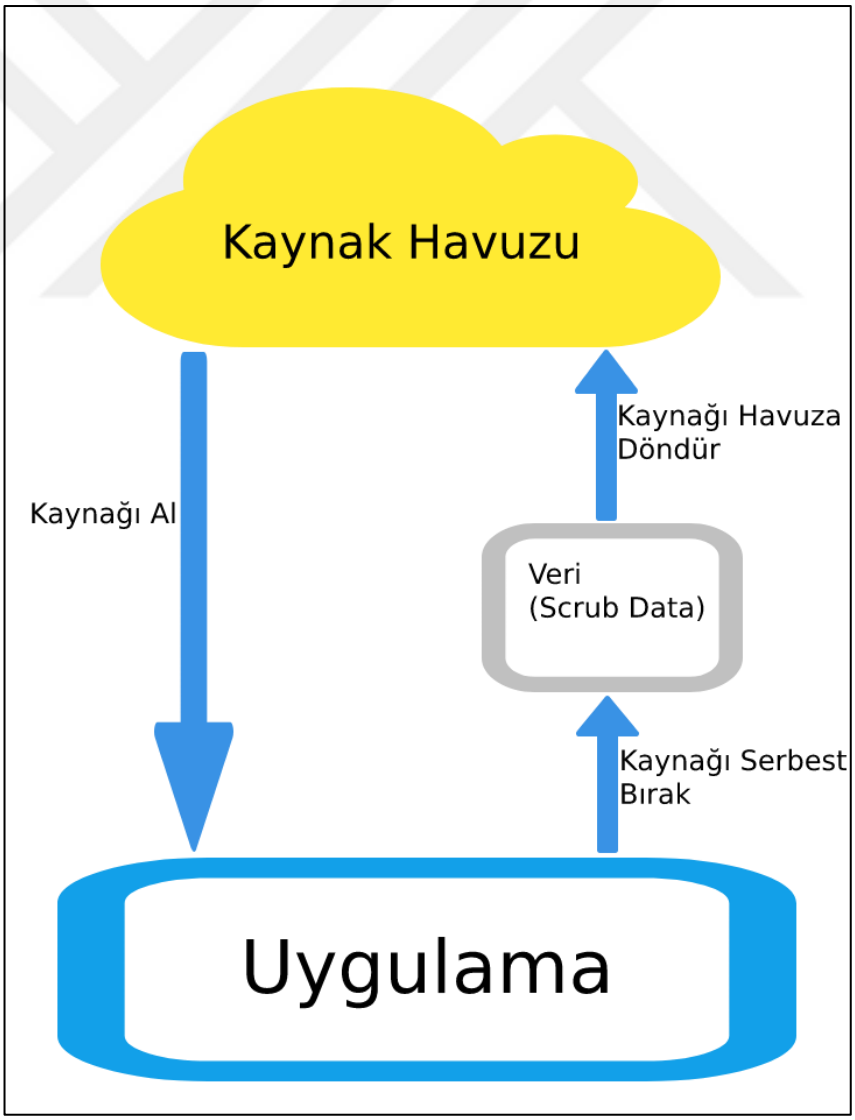
EncFS, Linux için bir dosya veya dizin düzeyince şifreleme sistemidir.

3.3.2 Hassas Bilgileri Temizle Tasarım Deseni

Yeniden kullanılabilir bir kaynakta depolanan hassas bilgilere, yeniden kullanılabilir kaynağı serbest bırakmadan önce hassas bilgilerin silinmemesi durumunda yetkisiz bir kullanıcı veya rakip tarafından erişilmesi mümkündür. Bu desenin kullanımı, kaynak yeniden kullanmadan önce hassas bilgilerin yeniden kullanılabilir kaynaklardan arındırılmasını sağlar. Bu güvenli tasarım deseni "yeniden kullanım için iade edilen yeniden kullanılabilir kaynaklarda depolanan hassas bilgileri temizle" CERT Güvenli Kodlama önerisini temel alır [22].

Hassas bilgileri temizle (Clear Sensitive Information) tasarım deseni, temizlenmesi gereken yeniden kullanılabilir bir kaynak aracılığıyla hassas bilgilere erişimi önler. Sistem, bellek, önbellek ve diskler gibi kaynaklar aracılığıyla hassas verilere erişebiliyorsa bu desen kullanılmalıdır. Bir kaynak serbest bırakıldığında, hala yetkisiz kullanıcılar tarafından erişilebilecek bilgilere sahip olabilir. Yeniden kullanılabilir kaynaklar dinamik olarak ayrılmış olan hafıza, statik olarak ayrılmış olan hafıza, otomatik olarak tahsis edilmiş bellek, hafıza önbellekleri, disk, disk önbellekleri gibidir.

Çoğu durumda, yeniden kullanılabilir bir kaynağı, dinamik olarak ayrılmış belleği boşaltmak veya bir dosyayı silmek gibi, kullanılabilir kaynak havuzuna döndüren eylem, kaynağı yeniden kullanım için uygun olarak işaretler. Yeniden kullanılabilir kaynağın mevcut içeriği, kaynak yeniden kullanılıncaya ve kaynağa yeni veriler yazılana kadar bozulmadan bırakılır. Bu yetkisiz bir kullanıcının serbest bırakılmış bir kaynakta geride bırakılan eski bilgilere erişerek potansiyel olarak hassas bilgi sızıntısına yol açabileceği anlamına gelir. Bu modelin ana odağı, yeniden kullanılabilir kaynağı mevcut kaynak havuzuna geri göndermeden önce tüm hassas verilerin silinmesini sağlamaktır. Şekil 3.14 hassas bilgileri temizle tasarım desenini göstermektedir.



Şekil 3.14. Hassas Bilgileri Temizle Tasarım Deseni Yapısı

Kaynak: Uygulama tarafından kullanılan yeniden kullanılabilir kaynak. Daha önce belirtildiği gibi, kaynak dinamik olarak ayrılmış bir bellek parçası, diskteki bir dosya olabilir.

Uygulama: Ana uygulama, yeniden kullanılabilir kaynak havuzundan alınan bir kaynağa hassas bilgileri yazar.

Kaynak Havuzu: Uygulama tarafından kullanılan yeniden kullanılabilir kaynakların kaynağını ifade eder.

Fırçalama-Sıkıştırma Verileri: Uygulama, kaynağı kaynak havuzuna döndürmeden önce, yeniden kullanılabilir kaynaktaki hassas bilgileri temizlemek için bazı mekanizmalardan faydalanmalıdır.

Uygulama tarafından kaynak havuzuna döndürülen yeniden kullanılabilir bir kaynağa erişim hakkı kazanan yetkisiz bir kullanıcı, hassas bilgileri okuyamaz.

3.3.3 Güvenli Dizin Tasarım Deseni

Güvenli Dizin (Secure Directory) tasarım deseninin amacı, bir saldırganın, programın çalışması sırasında program tarafından kullanılan dosyaları değiştirmesini önlemektir. Bir program, program yürütme sırasında bir süre için bir dosyaya bağlı olabilir. Program geliştiricileri, genellikle program tarafından kullanılan dosyaların, programın yürütülmesi sırasında dış kullanıcılar tarafından kullanılmayacağını varsaymaktadır. Bununla birlikte, bu varsayımın yanlış olması durumunda, bir dosya birden fazla kullanıcı tarafından değiştirilebilir; bu kötü niyetli bir kullanıcının programın değiştirilmemiş halde kalmasına bağlı olarak programda bir yarış durumuna neden olması nedeniyle kritik bir süre boyunca dosyayı değiştirebileceği veya silebileceği anlamına gelir. Güvenli dizin tasarım deseni, program tarafından kullanılan dosyaların depolandığı dizinlerin yalnızca program kullanıcısı tarafından yazılmasını sağlar.

Bu tasarım deseninin uygulanabilmesi için öncelikle dosyanın dizinindeki kanonik yol adı bulunur. Daha sonra kurallı yol adı tarafından referans verildiği şekilde dizinin güvenli olup olmadığı kontrol edilir. Güvenli dizin yapısı, dizinin kullanıcı ve süper kullanıcı ile sınırlı yazma izinlerine sahip olmasını sağlar. Diğer kullanıcılar güvenli

dizindeki dosyaları değiştiremez. Ayrıca, ilgilenilen dizinden önce görünen tüm dizinlerin, diğer kullanıcıların güvenli dizindeki dosyaları değiştiremez ve silemez.

3.3.4 Yol Adı Kanonikleştirme Tasarım Deseni

Yol adı kanonikleştirme (Pathname Canonization) tasarım deseninin amacı, bir program tarafından okunan veya yazılmış tüm dosyaların herhangi bir sembolik bağlantı veya kısa yol içermeyen geçerli bir yoldan, yani kanonik bir yoldan kaynaklanmasını sağlamaktır. Verilen dizinlere giden yolun geçerli olmasını ve diğer konumlarla bağlantı kurmaması sağlanır. Geliştirilen bir uygulama eğer kullanıcıdan yol adları kabul ediyorsa bu desen kullanılmalıdır. Yol adı kanonikleştirme tasarım deseni dizinde gezinme güvenlik açıklarına karşı koruma sağlar. Sembolik bağlar ve diğer dosya sistemi özellikleri nedeniyle, bir dosya aslında yolla gösterilen dizinde bulunmayabilir. Bu nedenle, yol adında dize temelli doğrulama yapılması yanlış sonuçlara neden olabilir. Doğru kurallı yol adının olması, güvenli olup olmadığını anlamak için bir dizini denetlerken özellikle önemlidir.

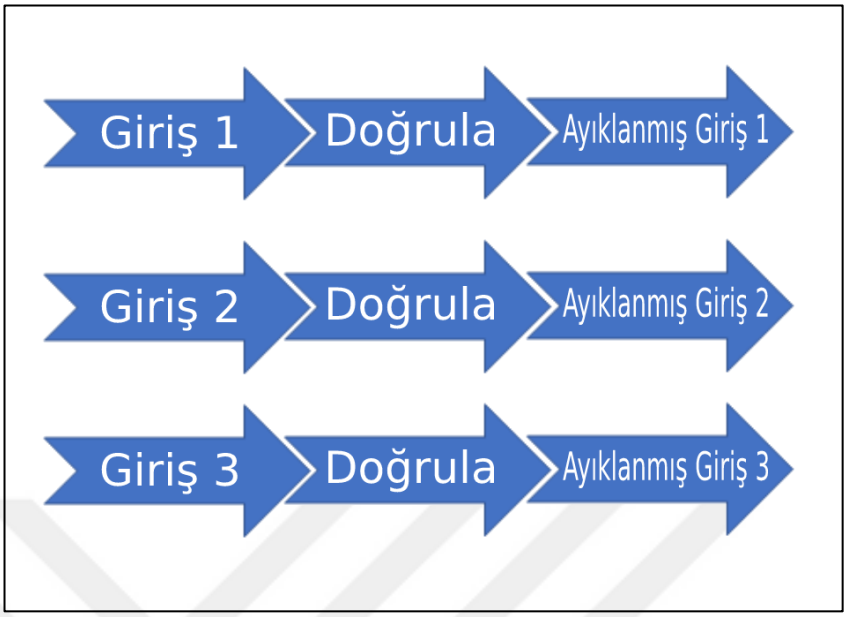
Programatik olarak yol adı kanonikleştirme tasarım deseninin yapısı, dosyayı açmadan önce verilen yoldaki işletim sistemi spesifik bir patika kanonikleştirme işlevinin çağrılmasını içerir. Kurallı yol adı dosya üzerinde çalışırken kullanılır. Kanonize edilmiş yol adının kendisi, sonuncu hariç kanonik yoldaki her öğe, orijinal dizin olacak ve bir bağlantı veya kısayol olmayacak şekilde bir yapıya sahiptir. Son öğe ise orijinal dosyanın adıdır.

3.3.5 Giriş Doğrulama Tasarım Deseni

Girdi verilerinin doğru şekilde doğrulanmasını sağlayarak birçok güvenlik açığı önlenebilir. Giriş Doğrulama, geliştiricilerin güvenilmeyen veri kaynaklarından gelen tüm harici girişleri doğru şekilde tanımlamasını ve doğrulamasını gerektirir. Giriş doğrulama (Input Verification) tasarım deseni, konunun girdiği verilerin geçerli olmasını ve kötü amaçlı metin içermemesini sağlar. Kullanıcı girişinin doğrulanamaması, sistemleri SQL enjeksiyon ve taşma (overflow) saldırıları gibi saldırılara maruz bırakabilir. Giriş Doğrulamayı uygulamak için geliştiricilerin hangi girişin güvenilmez olduğunu tanımlamaları ve kullanmadan önce bu girişi doğrulamaları gerekir. Giriş güvenilir sunucularda doğrulanmalıdır, ancak sunucu ile iletişimi en aza indirmek için istemci tarafında da yapılabilir.

Bir uygulama tarafından doğrulanmamış kullanıcı girdisinin kullanılması, arabellek taşması saldırıları, SQL enjeksiyon saldırıları ve siteler arası komut dosyası çalıştırma saldırıları gibi birçok ciddi güvenlik açığının kök nedenidir. İstemci sunucu mimarisine sahip uygulamaların yaygınlığı göz önüne alındığında, sistem tasarımcılarının karşılaştığı sorunlardan biri, girdi doğrulamasının istemci tarafında veya sunucu tarafında gerçekleştirilmesidir. Giriş doğrulamadaki sorunlar yalnızca istemci tarafında doğrulama yapıldığında meydana gelir. İstemci tarafı doğrulamaları doğal olarak güvensizdir. Bir web sayfası gönderimini taklit etmek ve orijinal sayfadaki herhangi bir komut dosyasını atlamak genel olarak kolaydır. Ancak, istemci tarafındaki doğrulama kullanışlı olmasına rağmen güvenilmemelidir. Anında kullanıcı geri bildirimi yapmak, sunucuya tekrar bir tur atılmasını önleyerek zamandan ve bant genişliğinden tasarruf sağlar.

Giriş Doğrulama tasarım deseni, güvenilmeyen bir kaynaktan veri kabul eden herhangi bir yazılım için geçerlidir. Güvenlik sınırı boyunca bir program arabirimine ulaşan herhangi bir veri doğrulama gerektirir. Bu tür verilerin genel örnekleri ortam, soketler, borular, dosyalar, sinyaller, paylaşılan hafıza ve cihazları içerir. Web uygulamalarına özgü bazı girdi kaynakları, http formlarından GET ve POST parametreleridir. Giriş Doğrulama modelinin yapısı oldukça basittir ve yalnızca şekilde 3.15'te gösterildiği gibi güvenilmeyen her girişin tanımlanması ve doğrulanması gerekir.



Şekil 3.15. Giriş Doğrulama Tasarım Deseni Yapısı

3.3.6 Kaynak Toplama Başlatma Tasarım Deseni

Kaynak toplama başlatma (Resource Allocation is Initialize) tasarım deseninin amacı, sistem kaynaklarının program yürütme yolları altında uygun şekilde paylaştırılmasını ve dağıtılmasını sağlamaktır. Kaynak edinme veya kaynak toplama başlatma tasarım deseni, bir nesnenin kaynaklarının istemciye güvenmek yerine yapıcı ve yıkıcı metotları kullanılarak doğru şekilde dağıtılmasını ve kaldırılması sağlar. Kaynak kullanımı, bellek yeniden tahsisini tetikleyecek kadar yoğun olduğu zamanlarda kullanılması gereken bir güvenlik tasarım desenidir. Bir sınıfı tekrar kullanmadan önce temizlenmesi gereken verileri ortaya çıkarabilen bu güvenlik tasarım deseni nesne tabanlı programlamada çöp toplayıcı (literatürdeki bilinen adıyla Garbage Collection) yöntemini çalıştırmaktadır.

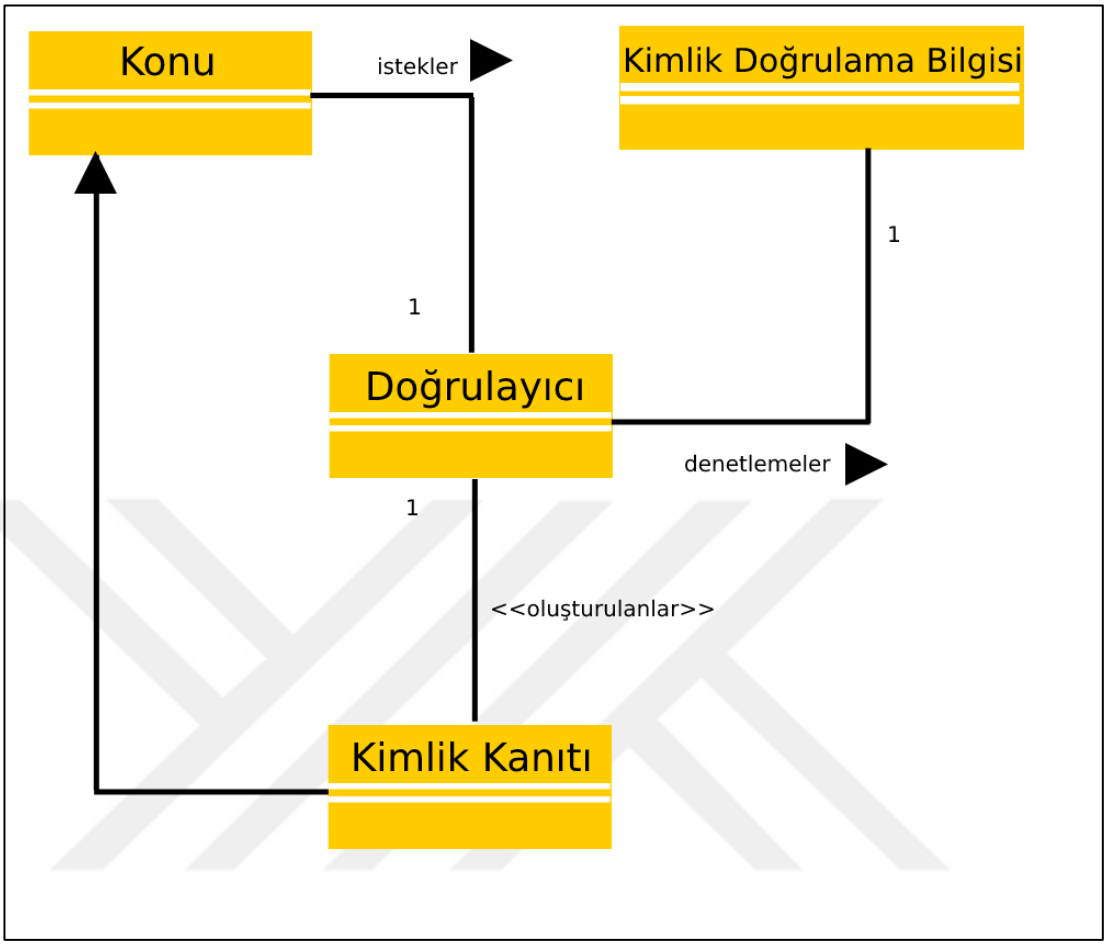
Yazılım geliştirmede kaynak kullanımını optimize edebilmek ve kaynak tüketimini önlemek için kullanılan kaynakların zamanında serbest bırakılması gerekmektedir. Bu optimizasyon işleminde henüz kullanılan kaynakların da bırakılmaması gerekmektedir. Nesne tabanlı programlama konsepti ile güvenlik tasarım deseni yaklaşımının ortak paydası ise zamanından önce serbest bırakılan hafızanın kullanımının güvenlik açığı olarak kabul edilmekte oluşudur. Bellek tahsis sisteminde

ne kadar çok hafıza talep edilirse sistem boş olan hafızayı yeniden kullanıma açacaktır. Bu kullanım ise orijinal hafıza serbest bırakıldıktan sonra yapıldığından, hafızaya kaydedilmiş verilerin üzerine yazılarak yapılmış olacaktır. Bu yüzden serbest kaynakların ne zaman kullanıma sunulacağı ile ilgili tasarım yazılım geliştirilmesi esnasında planlanmalıdır. Yazılım geliştirme sürecinde bir kaynağın ne zaman gerekli olacağını ve ne zaman serbest bırakılacağını kestirmek zor olacağından kaynak toplama başlatma güvenlik tasarım deseni oldukça önemli bir desen durumundadır.

Kaynak toplama başlatma güvenlik tasarım deseninin kullanımına örnek vermek gerekirse program içerisindeki bir işlev hataya düşerse veya bir şekilde programı durdurursa program yine de öncelikli olarak belleği boşaltacaktır. Yani bir fonksiyonun başlangıcında hafızayı tahsis eden ve fonksiyon sona ermeden hafızayı serbest bırakan bir program çalışır.

3.3.7 Kimlik Doğrulayıcı Tasarım Deseni

Kimlik doğrulayıcı (Authentication) tasarım deseni, bir kişinin iddia ettiği kişi olduğunu doğrular. Bu güvenli tasarım deseni, farklı kullanıcıları barındırmak için farklı kimlik doğrulama algoritmalarının kullanılmasını desteklemektedir. Kimlik doğrulayıcı tasarım deseni, güvenliği artırmak için kimlik doğrulama bilgilerini karmaşık bir yapı ile sağlar. Doğrulayıcı, hassas bilgileri güvenli bir alanda saklarken, farklı doğrulama algoritmaları gerektiren birçok kullanıcı türüne de cevap verebilmektedir. Algoritma karmaşıklığı, kullanıcının bildiklerinin kombinasyonlarına dayanan protokollerin kullanımına uyum sağlamak için değiştirilebilir. Kullanıcının ne türde olduğu, kullanıcının nerede olduğu ve sahip olduğu roller gibi kimlik doğrulama protokollerinin eklenmesi işlemin yürütme süresini ve karmaşıklığı artıracaktır.

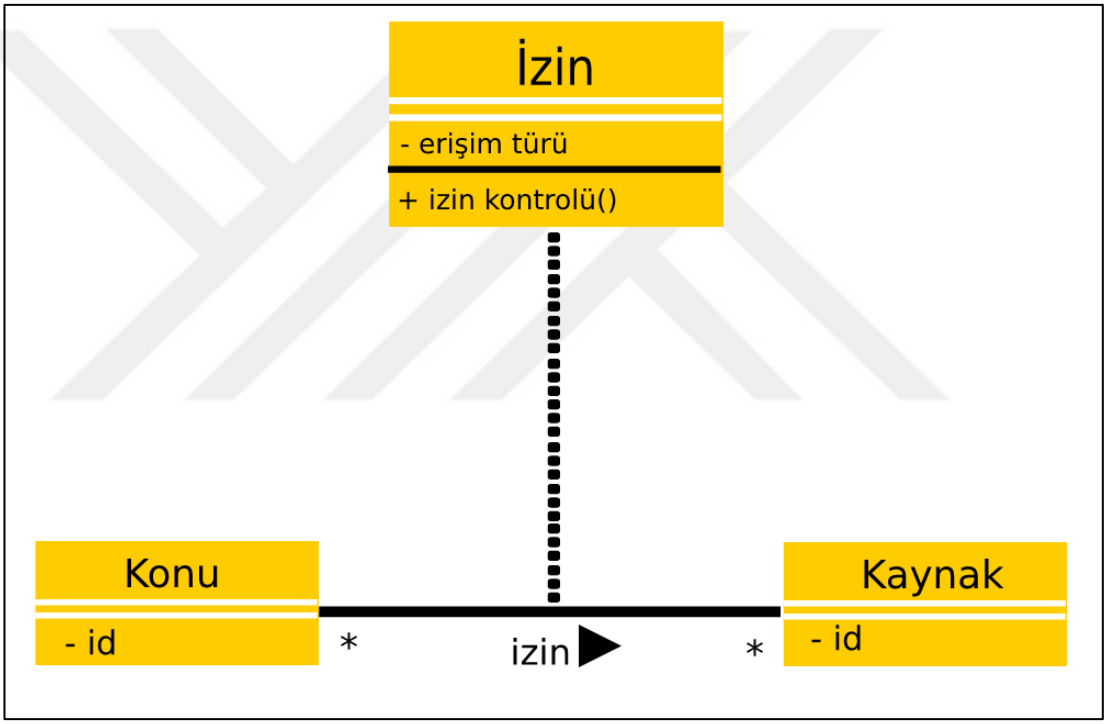


Şekil 3.16. Kimlik Doğrulayıcı Tasarım Deseni Yapısı

Şekil 3.16'da konu, kimlik kanıtı, doğrulayıcı ve doğrulama bilgisi katılımcıları ile kimlik doğrulayıcı tasarım deseni gösterilmektedir. Konu, istenen kaynaklara erişim verilmeden önce doğrulanması gereken varlıktır. Kimlik kanıtı, bir kez doğrulandıktan sonra konuya verilen nesne veya jetondur. Kimlik Doğrulayıcı, Konuyu ve Kimlik Kanıtını oluşturanın kimliğini doğrulamak için Kimlik Doğrulama bilgilerinin kullanan algoritmayı içeren nesnedir.

3.3.8 Yetkilendirme Tasarım Deseni

Yetkilendirme (Delegation) tasarım deseni, belirli bir kaynağa erişebilecek konuları ve aynı zamanda o kaynağa ait erişim haklarını tanımlar. [23] Bu tasarım deseni, izinleri konu ve kaynaklardan ayırır. Çeşitli konular, kaynaklar ve ayrıcalıklarla ilgilenmek için yeterlidir. Bu yetkilendirme kuralları bir konu ve kaynakla ilişkilendirileceğinden güvenlik politikaları, yetkilendirme kurallarına dahil edilecek konular için imtiyaz belirleme ayarlarını belirlemelidir.



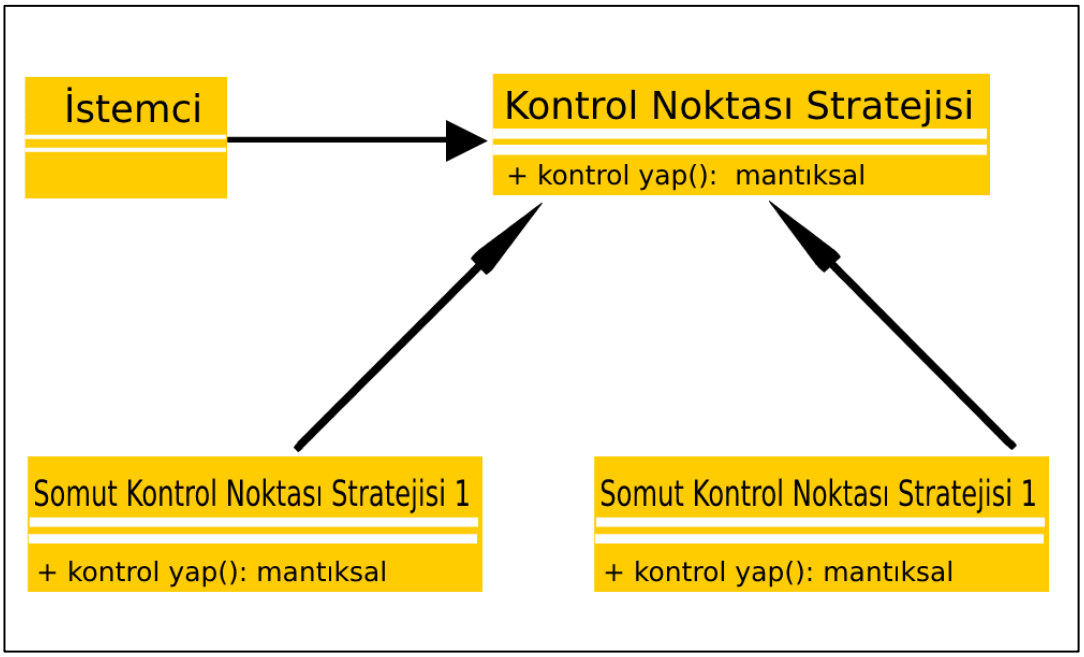
Şekil 3.17. Yetkilendirme Tasarım Deseni Yapısı

Şekil 3.17'de Konu, Görev ve Kaynak katılımcılarıyla Yetkilendirme Güvenli Tasarım Deseni gösterilmektedir. Burada Konu, kaynağa erişmesi gereken varlıktır. Her bir İzin, Konunun Kaynağa erişebildiği bir yolu temsil eder ve erişimi kontrol etmek için mekanizmalar sağlar.

3.3.9 Kontrol Noktası Tasarım Deseni

Kontrol noktası (Check Point) tasarım deseni, kullanıcıların gizli bilgilere erişmesini önler ve sistemin veri tabanına karşı kötü niyetli davranışlara koruma sağlar.
Kontrol noktaları, bir işletmenin güvenlik politikasını içine alan varlıklardır. Gamma' da [20] açıklanan güvenli strateji fabrikası deseni kullanılarak kontrol noktası deseni uygulanabilir.
Güvenli strateji fabrikası tasarım deseninin bu kullanımı, bir sistemin çalışmasının farklı noktalarına farklı kontrol mekanizmalarının uygulanmasına izin vererek güvenlik önlemlerinin müşteriye göre değişmesine izin verir. Bir kontrol noktası örneği, sisteme giriş yapan bir kullanıcı olabilir.

Kontrol noktası güvenli tasarım deseni, bir kullanıcının sisteme nasıl giriş yaptığı, kaç tane başarısız oturum açma girişiminin gerçekleştiğine veya günün saatlerine göre değiştirilebilir. Bu tasarım deseni, karmaşık güvenlik kontrollerini değiştirmeyi kolaylaştırmak ve uygulama durumuna bağlı olarak farklı stratejilerin kullanılmasına izin vermek için tek bir sınıfa ayırır. Bir kullanıcının kimliğini doğrulamak veya yetkilendirmek için kullanılabilir. Geçerli kullanıcı kimliği gibi genel bilgileri ayarlamak için kontrol noktaları kullanılabilir. [19]



Şekil 3.18. Kontrol Noktası Güvenli Tasarım Deseni Yapısı

Şekil 3.18 bir katılımcının kontrol noktası olduğu gösterilmektedir. Kontrol noktası, bir programın çalışma süresi boyunca herhangi bir zamanda kullanılmak üzere bir işletmenin güvenlik politikasını kapsar. Bu güvenli tasarım deseni, güvenlik mantığını normal uygulama mantığından ayırır ve güvenlik kontrollerinin değiştirilebilir ve yeniden kullanılabilir olmasını sağlar.

3.3.10 Bilgi Gizliliği Tasarım Deseni

Bilgi gizliliği (information privacy) tasarım deseni, verileri hırsızlıktan korumak amacıyla güvenli olmayan ortamlarda hassas verileri şifreler. Bu desen, veriler iç veya dış sistemler arasında sık sık geçirildiğinde kullanılmalıdır. Güvenlik politikaları, hangi verilerin şifreleneceğini ve bu verilerin ne kadar karmaşık bir şekilde şifreleneceğini belirlemek için verilerin hassasiyet düzeyini tanımlamalıdır. Şifreleme ve şifre çözme zaman alabildiğinden, mümkün olan yerlerde kullanımlarını sınırlamak en iyisidir. Bu örüntü, şifreleme ile nelerin gizlenmesi gerektiğini belirlemek için hassasiyet seviyesini kullanarak bu sorunu giderir.

Bu güvenli tasarım deseni, anahtar saklama mekanizmaları, şifreleme mekanizmaları ve şifreleme anahtarları ile algoritmalarını depolamak için güvenli bir konum gerektirir. Bu deseni kullanırken, hassasiyetini belirlemek için veriler sınıflandırılmalıdır. Bu sınıflandırma, hangi içeriğin şifrelenmesi gerektiğini belirlemek amacıyla kullanılmalıdır.

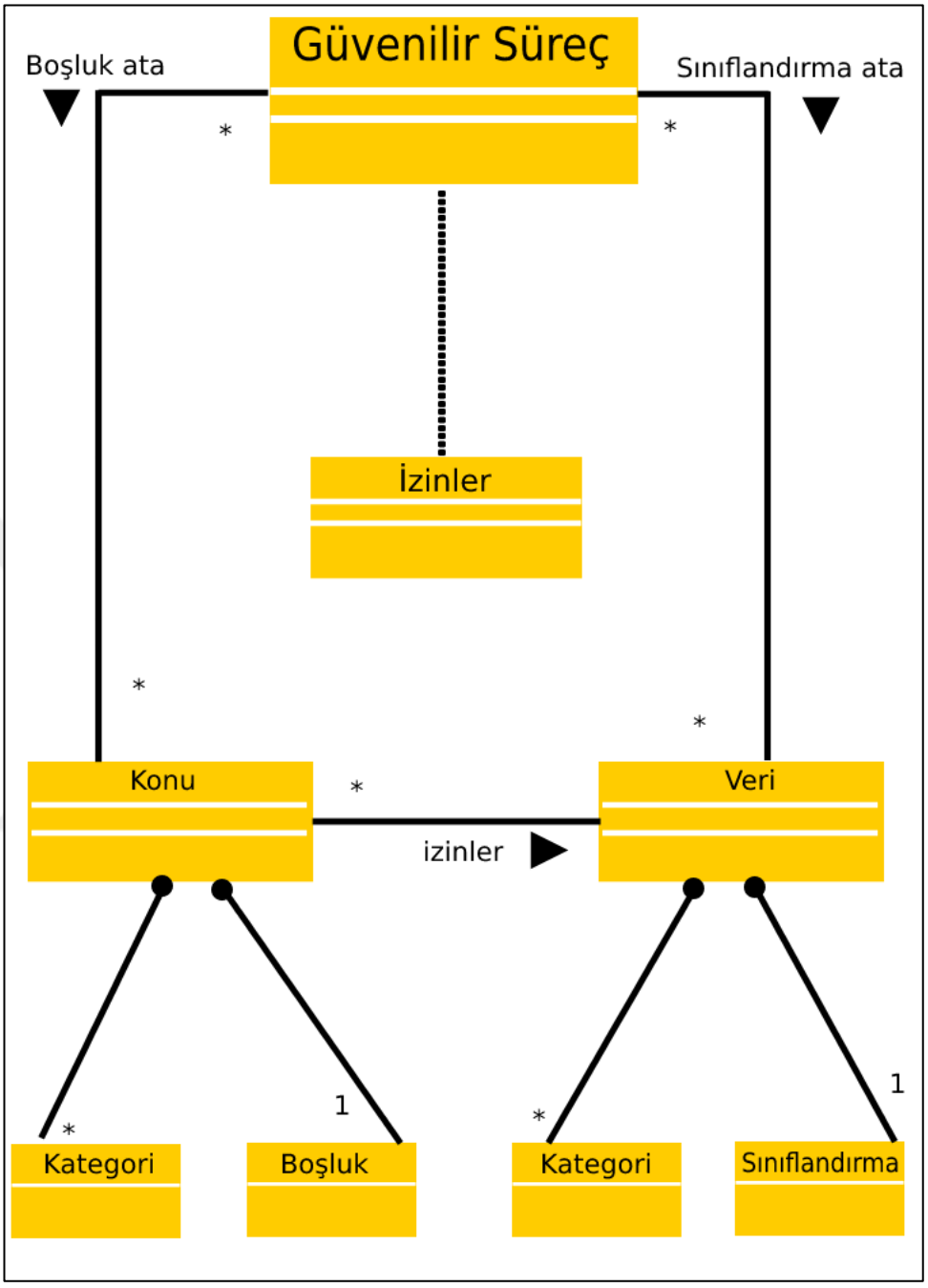
3.3.11 Sınırlı Görünüm Tasarım Deseni

Sınırlı görünüm (Limited View) tasarım deseni, yalnızca geçerli kullanıcının erişim için ayrıcalıklı olduğu işlevleri etkinleştirir. Bu desen, çoğu kullanıcı sınırlı sayıda işleme izin verdiğinde kullanılmalıdır. Bu desen kullanılırken, görünümü oluşturmadan önce güvenlik kontrolleri yapılır. Sınırlı görünüm, mevcut oturumun kullanıcı izinlerini kullanır ve mevcut kullanıcıya özel bir kullanıcı ara yüzü oluşturur. Sınırlı görünüm tasarım deseni, kullanıcı ara yüzünü oluşturmak için yapıcı metotlar kullanarak veya farklı görünümleri temsil etmek için bir durum makinesi kullanarak gerçekleştirilebilir. Sınırlı görünüm tasarım deseninin kullanımı, daha az güvenlik kontrolü ve daha az karmaşık bir kullanıcı ara yüzü ile daha temiz bir tasarımla sonuçlanır.

3.3.12 Çok Düzeyli Güvenlik Tasarım Deseni

Çok düzeyli güvenlik (Multi-level Security) tasarım deseni, kullanıcıları ve verileri kabul edilebilir kullanım modellerine göre kategorilere ayırarak veri nesneleri için erişim izinlerini belirler. İhtiyaç duyulan bu sınıflamaları değiştirmek için güvenilir süreçleri kullanır. Kullanıcılar için sınıflandırmalar boşluklar, veriler için sınıflandırmalar ise duyarlılık seviyeleri olarak adlandırılır.

Sınıflandırmaların kullanıcılar ve veriler için kullanılması sistemin gizliliğini ve bütünlüğünü korur. Bu desenin kullanımı, izinlerin bir kuruluş içindeki rütbesine veya konumuna dayalı olduğu kuruluşlarla sınırlı olmalıdır.



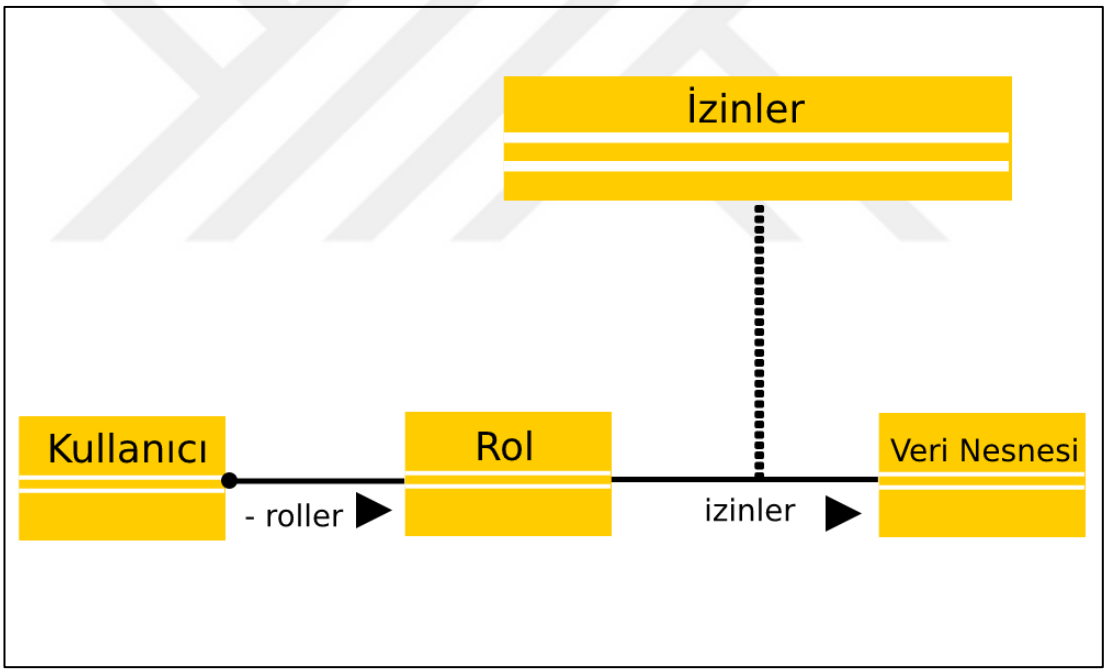
Şekil 3.19. Çok Düzeyli Güvenlik Tasarım Deseni Yapısı

Şekil 3.19'da yedi katılımlı ile Çok Düzeyli Güvenlik deseni gösterilmektedir. Katılımcılar, Konu, Veri, Güvenilir Süreç, Kategori, Gümrükleme, Kategori ve Sınıflandırma şeklindedir. Konu, ait olduğu organizasyon birimi için Kategori ve Gümrükleme' yi içerir. Açıklık, Konunun açıklık seviyesini temsil eder ve erişim izinlerini belirlemek için kullanılır. Veriler ayrıca ait olduğu organizasyon birimi için

Kategoriyi ve Sınıflandırmayı da içerir. Sınıflandırma Verilerin hassasiyet seviyesini gösterir ve erişim izinlerini belirlemek için kullanılır. Güvenilir Süreç, Verilerin Konu ve Sınıflandırma düzeyinin Açıklık seviyesini değiştirebilen varlıktır.

3.3.13 Rol Tabanlı Erişim Kontrolü Tasarım Deseni

Role tabanlı erişim kontrolü (Role Based) tasarım deseni, sisteme atanan rollere dayanarak izinleri kullanıcılarla ilişkilendirir. Kullanıcıyı rollerle ve izinlerle ilişkilendirme, bireysel kullanıcıları ayrı izin gruplarıyla ilişkilendirme çalışmasını ortadan kaldırır. Bu desen, çok sayıda kullanıcı ile çok sayıda kaynağın ilgili erişim ayrıcalıklarını paylaştığında uygundur.



Şekil 3.20. Rol Tabanlı Erişim Kontrolü Tasarım Deseni Yapısı

Şekil 3.20'de dört adet katılımcı ile Rol Tabanlı Erişim Güvenlik Tasarım Deseni gösterilmektedir. Desenin yapısındaki katılımcılar Kullanıcı, Rol, İzinler ve Veri Nesnesi şeklindedir.

Kullanıcı, sistemin geçerli kullanıcısını temsil eden sınıftır. Rol, kullanıcının üstlendiği mevcut rolü temsil eder. Veri Nesnesi, kullanıcın erişmek istediği verileri

temsil eder. Bir ilişkilendirme sınıfı, İzinler, rolün Veri Nesnesi için izinlerini belirler. Kullanıcının, Rolle çoktan çoğa bir ilişkisi olduğunu ve Rolün Veri Nesnesi ile çoktan çoğa ilişkisi olduğu unutulmamalıdır.
Bu desen, kullanıcıların yönetimi ve iş güvenliği için iş yükünü azaltır çünkü rollerden çok daha fazla kullanıcı vardır. Bu desen aynı zamanda kullanıcıların roller arasında geçiş yapmasını ve uygulamanın kullanımını kolaylaştırır.

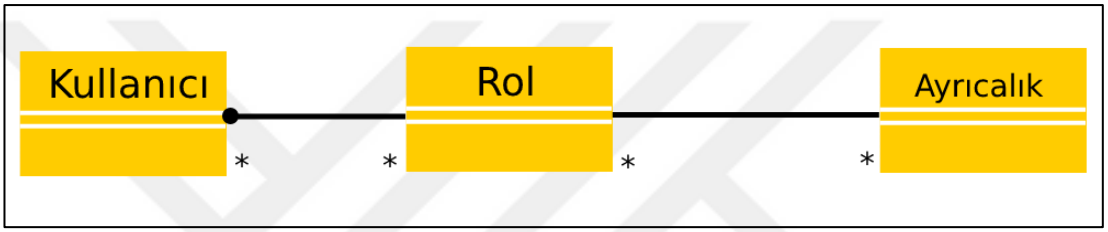
3.3.14 Rol Hakları Tanımı Tasarım Deseni

Rol hakları tanımı (Role Definition) tasarım deseni, kullanım durumlarına ilişkin izinlerin verilmesine dayanarak "en az imtiyaz" ilkesini benimsemektedir. Bu desen, roller işlevsel görevlere karşılık geldiğinde kullanılmalı, "en az ayrıcalık" prensibini kullanarak izinler verilmelidir. Bu desende, bir rolün bir sistemde hangi işlemleri gerçekleştirebileceğini belirlemek için sıra diyagramları kullanılır. Bu işlemler daha sonra bir rolün izinlerini belirlemek için kullanılır.
Rol hakları tanımı, rol tabanlı erişim kontrolü gibi diğer rol tabanlı izin desenleriyle birlikte kullanılır. Bu desen, sorumluluklarına göre rollerin asgari izinlerini belirlemek için kullanım durumunun geliştirilmesini ve dizi diyagramlarının analizini gerektirir.

3.3.15 Roller Tasarım Deseni

Roller (Roles) tasarım deseni, birden fazla kullanıcı için geçerli olan güvenlik politikalarını tek bir varlıkta birleştirir. Bireysel kullanıcılar yerine ayrıcalıkların rollerle ilişkilendirilmesi, ortak ayrıcalıklara sahip kullanıcı gruplarını yönetmeyi kolaylaştırır. Rolleri uygulama stratejisi, kullanıcının birden fazla rolü olmasına ve birden fazla ayrıcalıklara sahip olması için bir role izin vermesidir. Bu uygulama, her rolün bir paydaşı temsil etmesine ve paydaşın uygun ayrıcalıklarını yakalamasına izin verir. Rol tabanlı kalıtım ilişkileri ayrıca rol hiyerarşisi oluşturmak için de kullanılabilir.

Roller güvenli tasarım deseninin bir sonucu, yöneticilerin, kullanıcı ayrıcalık ilişkileri yerine kullanıcı rolü ve rol ayrıcalık ilişkilerini yönetmeleridir. Birden çok kullanıcıyı bir rol ile ilişkilendirmek, kullanıcıları kişiselleştirilmiş ayrıcalık kümeleriyle ilişkilendirmekten daha kolaydır. Diğer sonuçlar arasında, sistem ayrıcalıklarını gruplandırma ve yönetme konularına uygundur. Bu durum sisteme fazladan karmaşıklık katma pahasına fayda getirmektedir. Rollerin iyi bilinen bir uygulaması UNIX' in sahip grup-diğer rol sistemidir. Roller ayrıca aktörler, gruplar ve profesyoneller olarak da bilinir.



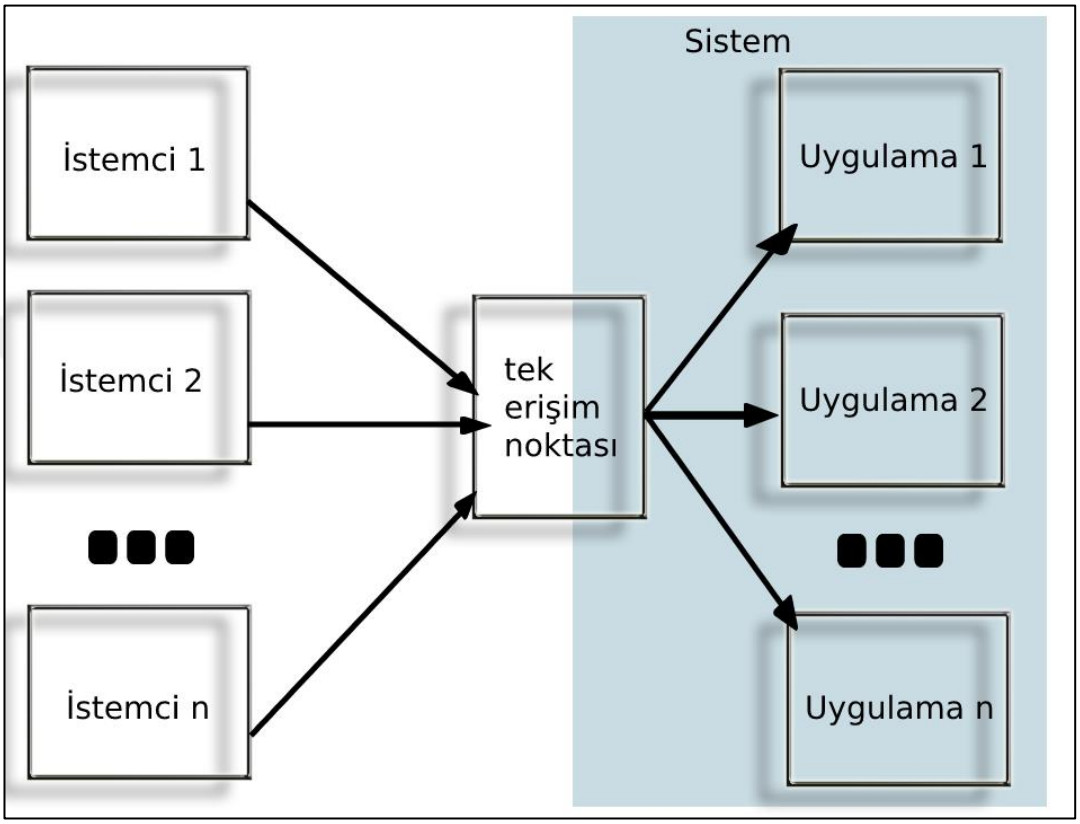
Şekil 3.21. Roller Tasarım Deseni Yapısı

Şekil 3.21'de üç adet katılımcı ile Roller Güvenli Tasarım Deseni gösterilmektedir. Katılımcılar Kullanıcı, Rol ve Ayrıcalık şeklindedir. Bir Kullanıcı, Rolle ilişkilidir. Kullanıcı bir Rolle ilişkilendirildiğinde, bu kullanıcı Rolle ilişkili Ayrıcalıklar kazanır.

3.3.16 Tek Erişim Noktası Tasarım Deseni

Tek erişim noktası (Single Access Point) tasarım deseni, bir uygulamaya erişimi bir giriş noktasına sınırlar. Güvenlik gereklilikleri, çoklu ara yüzleri destekleme ihtiyacı nedeniyle karmaşıklaştırmaktadır. Bu desen, kullanıcıları birden fazla giriş noktasında doğrulama ve bir uygulama boyunca güvenlikle ilgili kodu çoğaltma ihtiyacını ortadan kaldırır. Desenin yalnız giriş noktası, sistemde ihtiyaç duyulan bir kullanıcı hakkında tüm bilgileri toplamaktadır. Giriş noktası ayrıca tüm alt uygulamaları başlatmalı ve kullanıcıyı mevcut oturum için gerekli olmayan bilgileri girmesi için zorlayabilir.
Bu desen, sistemde bir kullanıcının oturum açması için kopyalanmış kodla sonuçlanabilecek diğer uygulamaların bir bileşimi olduğunda kullanılabilir. Birden fazla giriş noktası, kullanıcının sistemin farklı bölümlerine erişirken gereksiz bilgileri sağlamasını gerektiren farklı verilere ihtiyaç duyabilir. Bu daha büyük bir saldırı

yüzeyiyle sonuçlanacağından, sistemi daha az güvenli hale getirir ve daha kolay kontrol akışı sağlayarak sistemin test edilmesini kolaylaştırır.



Şekil 3.22. Tek Erişim Noktası Tasarım Deseni Yapısı

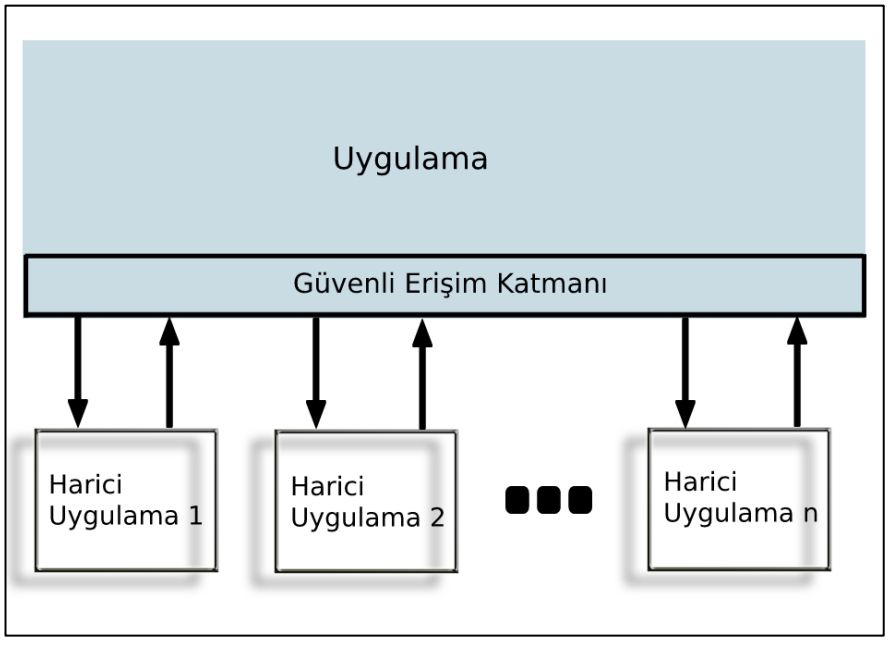
Şekil 3.22'de Tek Erişim Noktası Güvenli Tasarım Deseni gösterilmektedir. Şekilde Müşteri, Tek Erişim Noktası, Sistem ve Uygulama şeklinde dört katılımcı bulunmaktadır. Müşteri, Sisteme erişmek isteyen kullanıcı veya süreçtir. Bu, Tek Erişim Noktasından geçerek yapılır. Sistem, Müşterinin erişmek istediği uygulamadan oluşur.

3.3.17 Güvenli Oturum Tasarım Deseni

Güvenli oturum (Secure Session) tasarım deseni, bir uygulamada kullanılmak üzere, kullanıcının kullanıcı adı ve rolleri gibi genel olarak alakalı bilgileri depolamak için kullanılabilir. Oturum, genellikle web uygulamalarında bir kullanıcının istekler arasında bilgilerini depolamak için kullanılır. HTTP durumsuz olduğundan, Oturum kullanmak bir kullanıcının erişim kimlik bilgilerini yeniden girmek zorunda kalmadan uygulamayla etkileşimde bulunmasını sağlar. Güvenli Oturum, tüm bileşenlere sunulan ortak verileri depolamak için bir konum sağlar.

3.3.18 Güvenli Erişim Katmanı Tasarım Deseni

Güvenli erişim katmanı (Secure Access) tasarım deseni, harici sistemlerle iletişim kurmak için güvenli bir yöntem sağlar. [24] Sistem entegrasyonu genellikle ilk sistemlerin Uygulama Programlama Ara Yüzleri (API) arasındaki sınırlarda zayıflıklar bırakır. Bu sınır noktalarının her iki tarafına da güvenlik kontrolleri uygulamak, entegre sistemi korumak ve test etmek için daha zor hale getiren yinelenen kodlar verebilir. Bu desen, güvenli olmayan harici sistemlerde arabirim yaparken kullanılmalıdır.



Şekil 3.23. Güvenli Erişim Katmanı Tasarım Deseni Yapısı

Şekil 3.23'de gösterilen güvenli erişim katmanı, farklı uygulamalar arasında iletişimi izole ederek sistemin bakımını ve testini kolaylaştırır. Güvenli Erişim Katmanı kullanılarak, bir sistem daha kolay bir şekilde farklı bir platforma taşınabilir.

3.3.19 Güvenli Kanallar Tasarım Deseni

Güvenli kanallar (Secure Channels) tasarım deseni, özellikle ortak ağlar üzerinden iletişim kurarken istemciler ve sunucular arasındaki bağlantıların güvenli olmasını sağlar. Güvenli kanallar tasarım deseni, şifrelemenin ek masrafını azaltmak için diğer verileri net bir şekilde iletirken hassas verileri şifreler. Bu desen, istemci ve sunucunun güvenli bağlantılar kurmasını gerektirir.

Bu deseni kullanmanın faydaları, hassas olmayan verilerin değişimini etkilemeden güvenli iletişim uygulamak için mevcut teknolojinin kullanılması nedeniyle geliştirilmiş güvenlik ve minimum geliştirme süresidir. Bu desenle ilgili sorunlar, şifreleme süresinden kaynaklanan düşük performans, müşteri iletişiminin ölçeklenebilirliği ve artan maliyettir.

4. ÖRNEK BİR UYGULAMA

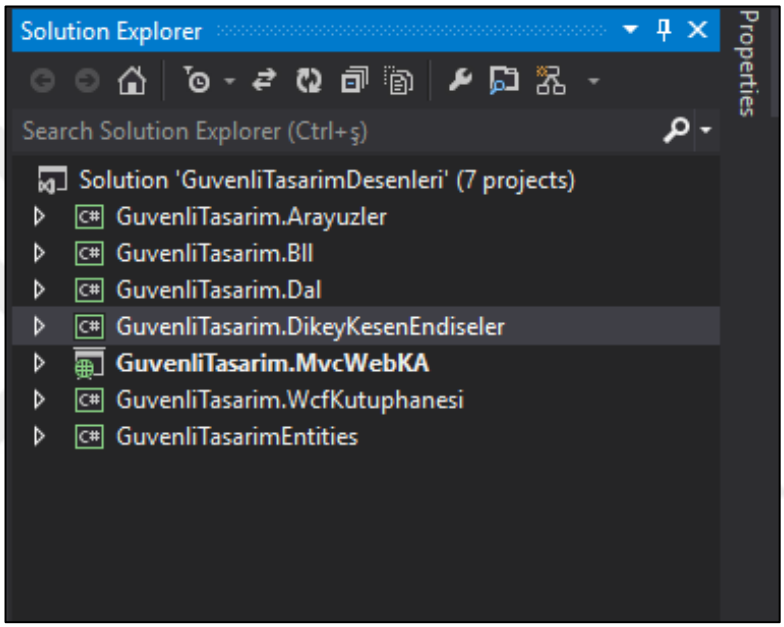
Örnek olarak geliştirilen uygulama Microsoft Asp.Net MVC platformu ile geliştirilmiş olup kaynak kodlama işlemi Microsoft C# .Net ile gerçekleştirilmiştir. Günümüzde sıklıkla son kullanıcı bazında kullandığımız veya geliştirici olarak kodlanan uygulamaların başında e-ticaret siteleri gelmektedir. Örnek uygulamada da bir e-ticaret sitesi senaryosu ile gidilmiştir. Tezin amacı yalnızca güvenli tasarım desenlerini belirgin hale getirmek olduğu için grafiksel tasarım ve veri tabanı üzerinde durulmamıştır. Veri tabanı olarak Microsoft'un geliştiricilere eğitim amacıyla ücretsiz olarak sunduğu hazır veri tabanlarından bir tanesi olan Northwind veri tabanı projeye dahil edilmiştir. Bu veri tabanı içerisinde ürün, müşteri, sipariş, stok gibi tablolar hazır olarak yer almaktadır. Ayrıca bu tabloların içerisinde hazır veriler de bulunmaktadır. Örnek proje bir e-ticaret sitesinin sepete ürün ekleme, sepetten ürün çıkarma, doğrulama kontrolleri gibi kodlamaların bulunduğu sınıflardaki güvenli tasarım desenlerinin uygulanışını göstermektedir.

Güvenli tasarım desenleri veya tasarım desenlerinin olduğu projelerin nesne tabanlı projeler olması kaçınılmazdır. Yüksek seviyeli kodlamaların yapıldığı özellikle ticari projelerin hemen hepsi nesne tabanlı programlama yaklaşımı ile geliştirilmektedir. Nesne tabanlı programlama (NTP), normal bir programlama sürecini çok daha hızlı bir şekilde yapmayı sağlayan, kodların sınıflar içerisinde yazıldığı ve bu sınıflardan türeyen nesneler üzerinden kodların çağrılması şeklinde devam eden bir yaklaşımdır. Nesne tabanlı veya nesne yönelimli programlama yaklaşımı ile katmanlı mimariler üzerinde geliştirme yapmak da oldukça yaygın bir geliştirme ortamı sunmaktadır. Tüm bu yaklaşımların üzerine bir de güvenlik tasarım desenleri ve tasarım desenleri geldiğinde ortaya daha profesyonel kodlar ve dolayısıyla da projeler çıkmaktadır.

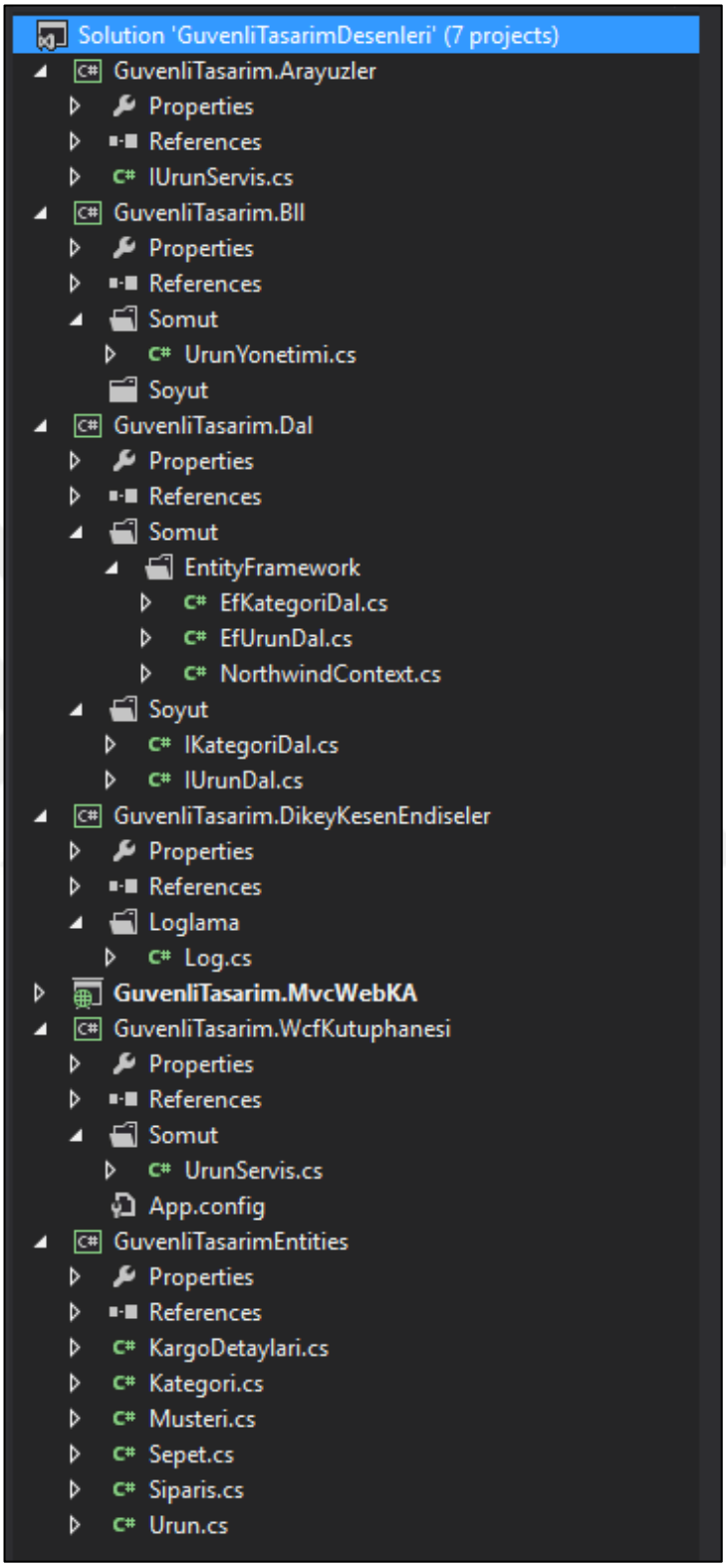
Örnek uygulamada MVC ile bir çözüm (solution) başlatılmış, bu çözüm içerisinde ise katmanlı mimariye uygun olması açısından birden fazla proje oluşturulmuştur. Oluşturulan MVC çözümünün yani örnek uygulamanın çatı isimlendirmesi GuvenliTasarimDesenleri şeklindedir. İçerisindeki projelerden bir tanesi veri erişim katmanıdır. Bu katmanın ismi kalıplaştığı için GuvenliTasarim.DAL (DAL: data access layer) şeklinde oluşturulmuştur. Örnek uygulamanın çözümüne yeni proje ekle yöntemi ile eklenen bu proje sınıf kütüphanesi (class library) türündedir. Çözüme eklenen projelerden bir diğeri ise GuvenliTasarim.BLL adıyla oluşturulan mantıksal iş katmanıdır (BLL: Business Logic Layer) fakat yine isimlendirmesi kalıplaştığından anlam kaybı olmaması için İngilizce olarak oluşturulmuştur. Projelerden bir diğeri ise sonradan yapılacak değişiklikleri güvenli bir şekilde yönetmek amacıyla oluşturulan GuvenliTasarim.Arayuz (Interface) projesidir.

Örnek uygulamada farklı makineler üzerindeki uygulamaların birbiriyle iletişim kurma ihtimali düşünülerek bu sürecinde güvenli bir şekilde sağlanabilmesi için Windows iletişim iç sistemine (WCF) örnek sınıf yazılmıştır. Bu proje ise GuvenliTasarim.WcfKutuphanesi olarak isimlendirilmiştir. Örnek uygulamanın kullanıcı ile etkileşime geçebilmesi amacıyla olmazsa olmazı web projesi ise GuvenliTasarim.MvcWebKA adı ile oluşturulmuştur. Bu isimlendirmenin sonundaki KA, kullanıcı arayüzü (UI) anlamında verilmiştir. MVC ve nesne tabanlı programlama yaklaşımı ile oluşturulan uygulamada veri tabanı işlemlerini gerçekleştirecek olan katman ise Varlıklar katmanıdır ve kalıplaşmış ismi nedeniyle uygulamaya GuvenliTasarimEntities ismiyle yeni bir proje olarak eklenmiştir. Bu projelerin her birinin içerisinde ise üzerine düşen görevleri yapacak sınıflar bulunmaktadır.

Tüm bu isimlendirmeler, sınıflar, aksiyonlar, metotlar, fonksiyonlar ve projeler yalnızca nesne tabanlı programlama yaklaşımına göre değil aynı zamanda yazılım güvenlik açıklarına karşı güvenli tasarım desenlerinin uygulanacağı bir yaklaşım ile oluşturulmuştur. Örnek uygulamada yer alan projeler Şekil 4.1'de onların altında bulunan sınıflara ait ekran görüntüsü ise Şekil 4.2'de verilmiştir.



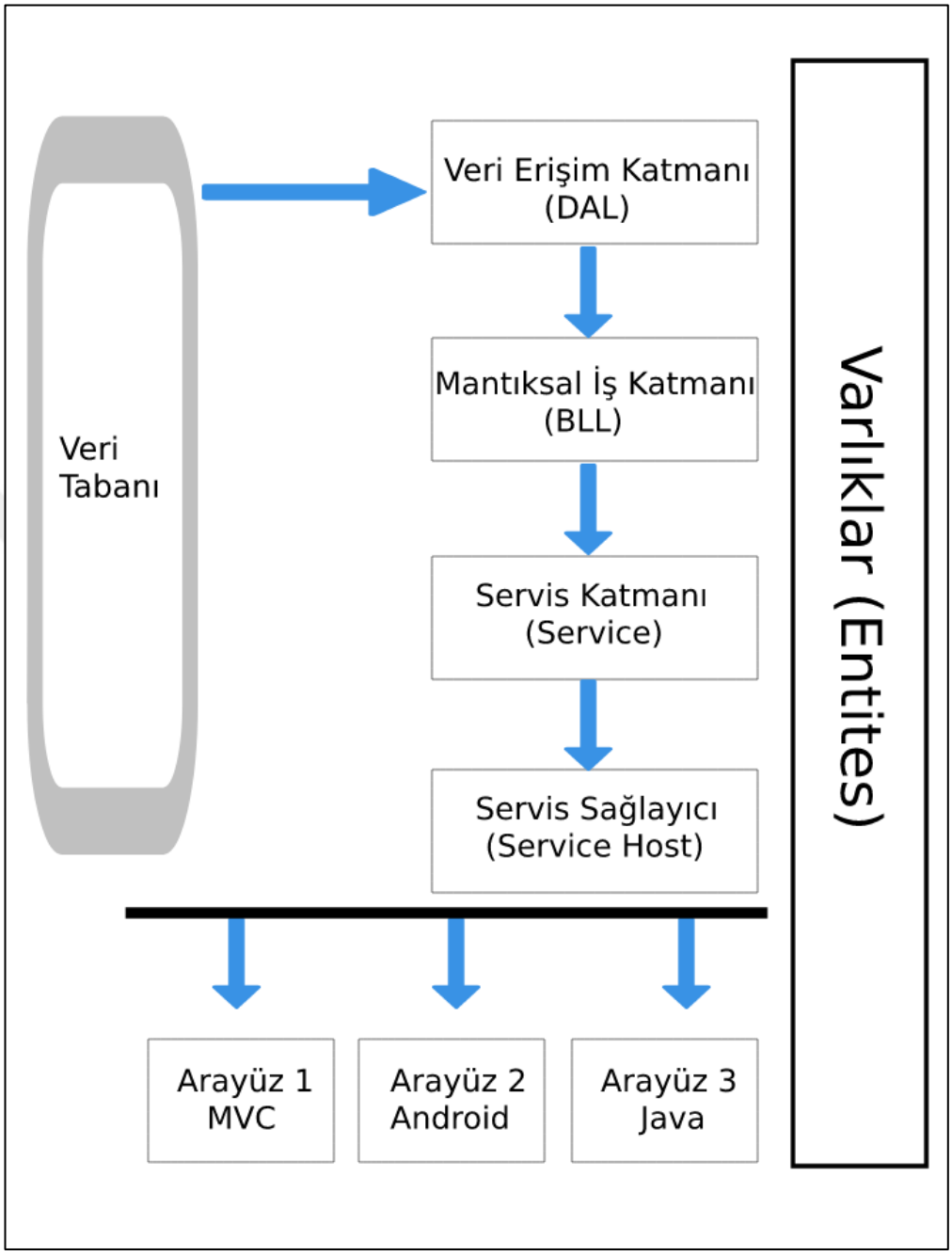
Şekil 4.1 Örnek Uygulama Çözüm Penceresi



Şekil 4.2 Örnek Uygulama Projelerin Görüntüsü

Örnek uygulamaya ait çözüm penceresinin ekran görüntülerinden de anlaşılacağı üzere kurgulanan programlama ortamının yapısı (backend) katmanlı mimari şeklindedir. Daha önce de bahsedildiği gibi güvenli tasarım deseni denilince yalnızca herhangi bir güvenli tasarım deseni ve ona ait çalışma yapısı veya kodlar akla gelmemelidir. Güvenli tasarım deseni bir disiplindir ve uygulamayı tümüyle çevrelemelidir. Bu yüzden örnek uygulamada da olduğu gibi katmanlı bir mimari ile çalışmak ayrıca bir de servis tabanlı geliştirme yapmak aslında bir güvenli tasarım deseni modelidir. Güvenli tasarım desenlerinin doğrudan uygulanmadığı bir katmanlı mimari bile diğer sistemlere nispeten güvenlidir. Veri erişim katmanı (DAL) altındaki IUrunDal somut Urun sınıfının soyutudur ve isimlendirmesinde I harfi ara yüzü (interface) ifade etmektedir.

Oluşturulan örnek uygulamadaki birkaç sınıfta güvenli tasarım desenleri ile geliştirme yapılmıştır ve bu geliştirme ortamı kodlarıyla birlikte açıklanacaktır fakat oluşturulan örnek uygulamanın ve mimarinin daha iyi anlaşılması için öncelikle aşağıda bulunan Şekil 4.3 incelenmelidir.

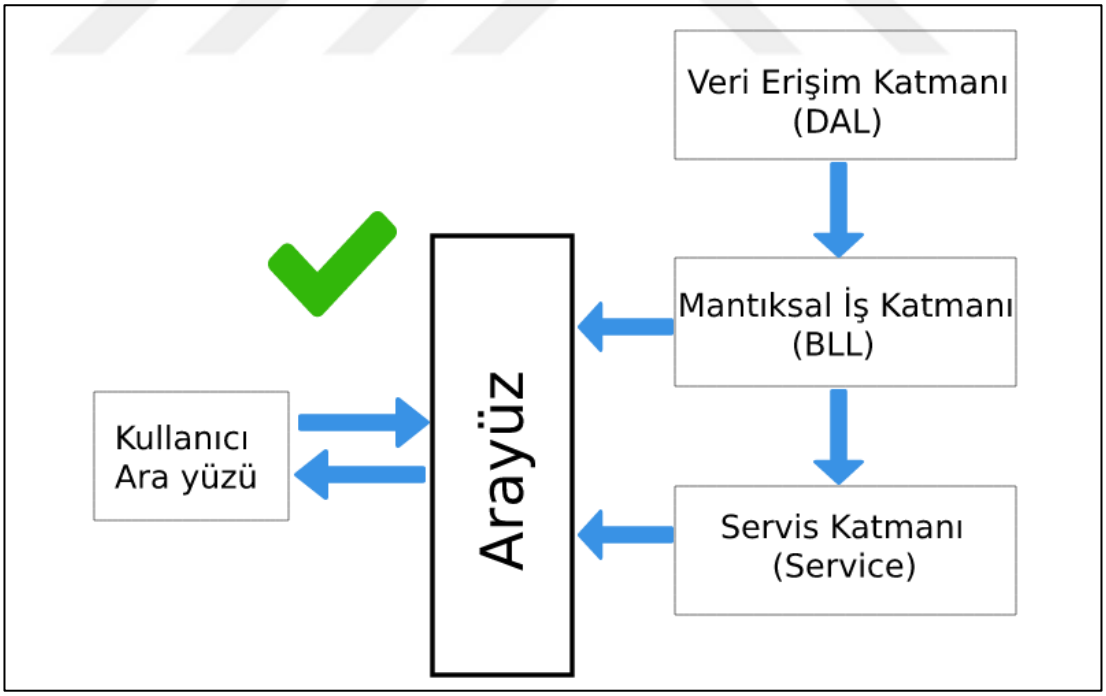


Şekil 4.3 Örnek Proje Güvenli Mimari Deseni

Örnek uygulamadaki çözüm (solution) içerisinde yer alan varlıklar projesi (entities) altında somut (concrete) sınıflar oluşturulmuştur. Bu sınıflar örnek bir e-ticaret senaryosu üzerinden gidildiği için sipariş, müşteri, ürün ve kategori şeklindedir. Güvenli tasarım desenleri ile yazılım geliştirmede benimsenmesi gereken bir yöntem de somut sınıflar oluşturmaktır. Veri erişim katmanında bulunan Soyut (Abstract)

klasörü altındaki sınıflar ise somut sınıfların soyutu olacak şekilde hazırlanmıştır. Veri tabanına erişim ve veri ile ilgili işlemler Microsoft Entity Framework ile gerçekleştirilmiştir. Entity Framework ile yapılan değişikliklerin veri tabanına kaydedilmesini sağlayan kod SaveChanges(); olarak bilinir ve aslında bu kodun arka planda, kütüphanede yaptığı iş işlemlerin arka arkaya veri tabanına işlenmesini sağlamaktır. Bu yöntemde iş birimi (Unit Of Work) olarak bilinen tasarım deseninin bir örneğidir.

Güvenli tasarım desenlerinin kullanılması ile yazılımda değişimin gerekli olduğu durumlar da güvenli bir şekilde yönetilmektedir. Soyutlama kavramı bu yüzden hem güvenli tasarım desenlerinde hem de nesne tabanlı programlama geliştirmede çok önemlidir. Bir yazılımda kullanıcı ara yüzü (UI) servis veya mantıksal iş katmanı (BLL) ile direk muhatap olabilir fakat güvenli tasarım deseni sayesinde kullanıcı ara yüzü, mantıksal iş katmanı ve servis ile doğrudan ilişki içerisinde olan ara yüz () ile muhatap olur.



Şekil 4.4 Kullanıcı Ara yüzünün Diğer Katmanlarla Doğrudan Çalışması Prensibi

Şekil 4.4'te gösterilen mimariyi gerçekleştirmek üzere örnek uygulamaya GuvenliTasarimDesenleri.Arayuzler projesi eklenmiştir. Projenin altında ise ara yüz ile ilgili servis işlemlerin gerçekleştirileceği (IUrunServis) sınıf eklenmiştir. Uygulamanın çatısını oluşturan MVC web projesi de aslında bir desendir. Bu desen Microsoft tarafından kullanışlı bir kütüphane haline getirilmiştir. Bir asp.net mvc kütüphanesi model, view ve contoller ara yüzlerinden oluşmaktadır. UrunContoller.cs içerisinde yer alan kodlar mvc deseni içerisinde bağımlılık enjeksiyonu tasarım deseni (dependency injection design pattern) için örnek gösterilebilir.

Güvenli tasarım desenlerinin MVC projelerinde genel bir tasarım olarak kullanılabilmesi için alt yapı niteliğinde bir klasör oluşturulur. Bu klasör içerisinde güvenli tasarım desenlerinin yorumlanması için gerekli kütüphane oluşturulur. Güvenli tasarım desenleri için alt yapı oluşturacak olan bu klasör içerisindeki sınıfa kontrol fabrikası ismi verilmiştir. Örneğin uygulama içerisinde bir kontrolör çalışırken, o kontrolörün yapıcı metodundan A servisi istenirse o çağrıya mantıksal iş katmanının verilmesi yorumunu katacak olan kontrolör alt yapısı bu fabrika içerisinde çalışacaktır. Kontrolör fabrikasını (Controller Factory) ayağa kaldıracak olan sınıf ise MVC kütüphanesinde zaten mevcut olan varsayılan kontrolör (Default Controller Factory) fabrikasıdır. Bunu kalıtım yoluyla miras almaktadır. Kod örneği, KontrolorFabrikasi:VarsayilanKontrolorFabrikasi şeklindedir.

Fakat kontrolör fabrikasının mevcut olarak yaptığı yorumlama bizim tercih ettiğimiz bir yorumlama olmadığından istediğimiz yorumlamayı yapabilmek amacıyla geçersiz kılma (override) metodunu kullanıyoruz. Bu işlemin gerçekleştirildiği kodlama şekil 4.5'te gösterilmiştir.

```
public class KontrolorFabrikasi:DefaultControllerFactory
{
private ICekirdek _kontrolorCekirdegi;
public KontrolorFabrikasi()
{
_kontrolorCekirdegi = new StandardCekirdek();
AddBllBindings();
}
private void AddBllBindings()
 {
_kontrolorCekirdegi.Bind<IUrunServis>().To<UrunYonetimi>().WithConstructorArgument("UrunDal", new EfUrunDal());
//eğer IUrunServis istenirse UrunYonetimi'ni ver işleminin yapıldığı yer.
 }
// yukarıdaki kodun bu şekilde yorumlanması gerektiği ise aşağıdaki kod ile tanımlanır.
protected override IKontrolor GetControllerInstance(RequestContext requestContext, Type controllerType)
{
    return controllerType == null ? null : (IKontrolor) _kontrolorCekirdegi.Get()(controllerType);
}
}
```

Şekil 4.5 Kontrolör Fabrikasına Ait Kodlama

MVC projeleri aslında varsayılan kontrolör fabrikasını (DefaultControllerFactory) zaten kullanmaktadır ama güvenli tasarım desenlerinin uygulanması için bu fabrikanın geçersiz kılma metodu ile bir adet kontrolör döndürüleceği zaman bu kontrolör döndürülürken bir yapıcı metoda ihtiyaç duyabileceğinde işlemi başarıyla gerçekleştirebilmesi sağlanmıştır.

E-ticaret sitelerinde üyelerin sitede geçirdikleri süre boyunca yaptıkları tüm işlemler oturum yönetimine ilişkin yöntemlerle tutulmaktadır. Sepete ürün eklenmesini içeren aşamada sepete eklenen ürünleri hafızada tutma işi veri tabanında değil oturum yönetiminde (Session) tutulmaktadır. Sepete ürün ekleme ve sepetten ürün çıkarma gibi işlemlerin güvenli tasarım desenleri kullanılarak güvenli bir şekilde geliştirildiği kodlama aşağıdaki gibidir. Kodlamaya geçmeden önce güvensiz bir sepet çalışma mekanizmasında kullanıcıların yanlış ürünü sipariş etme, ürünü kaldırmama veya ürünü almaktan vazgeçmesine rağmen ürün için ödeme yapılması gibi olumsuz sonuçlar doğacaktır. Ayrıca iyi kurgulanmamış bir sepet modülü kullanıcıların ödeme bilgilerinin çalınmasına yol açabilir.

Sepet işlemlerinin varlıklar (entities) projesi altında sepet sınıfına ait kodlaması şekil 4.6 ve 4.7'de gösterilmektedir.

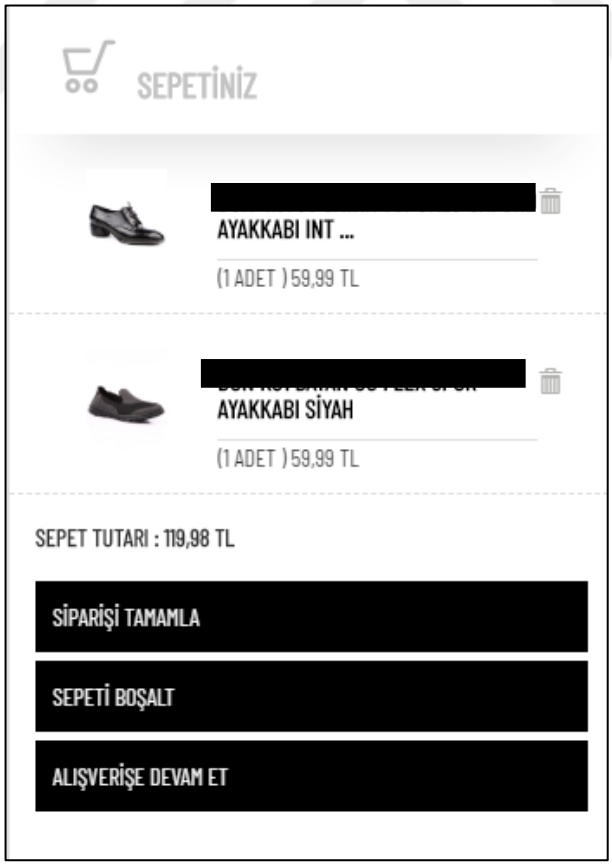
```
namespace GuvenliTasarimDesenleri.Entities
{
    public class Sepet
    {
        List<SepetUrunSayisi> _sayi = new List< SepetUrunSayisi >();
         public void SepeteEkle(Urun urun, int miktar)
    {

SepetUrunSayisi sepeturunsayisi = _sayi.FirstOrDefault(c => c.Urun.UrunID== urun.UrunID);

 // ürün var mı diye kontrol ediliyor.

             if (SepetUrunSayisi ==null) // ürün yoksa sepete ürün ekleniyor
             {
                 _sayi.Add(new SepetUrunSayisi {Urun=urun,Miktar=miktar});
              }

            else
            // ürün varsa ürün sayısı parametrede gönderilen ürün sayısı kadar artar.
             {
                 SepetUrunSayisi.Miktar += miktar;
             }
        }

```

Şekil 4.6 Sepet İşlemlerinin Güvenli Denetiminin Uygulanması

Şekil 4.7 Sepete Ekle Ekran Görüntüsü

```
public void UrunCikar(Urun urun) // sepetten ürün çıkartma
        {
            _sayi.RemoveAll(p => p.Urun.UrunID == urun.UrunID);
        }
        public decimal Toplam  // sepetteki ürünlerin toplam fiyatı
        {
            get
            {
                return _sayi.Sum(c => c.Urun.BirimFiyat * c.Miktar);
            }
        }
public void Temizle()  //sepetteki tüm ürünleri temizleme
        {
            _sayi.Clear();
        }
    }
public class CartLine
    {
        public Urun Product { get; set; }
        public int Quantity { get; set; }
    }
}
```

Şekil 4.8 Sepet İşlemlerinde Güvenlik Kontrolünün Uygulanması



Şekil 4.9 Sepetteki Ürünler ve Ürün Çıkarma Ekran Görüntüsü

Sepet işlemlerinin MVC Web projesi altındaki kontrolöre (SepetContoller) ait kodları şekil 4.10 ve 4.11'de gösterildiği gibidir. Güvenli tasarım desenine uygun olarak ekleme işlemi bu kodlamada gösterilmektedir. _sepetServis'de urunId'si verilen ürünler listelenmektedir. Session'da sepet diye bir nesne var mı diye kontrol edilir. Eğer Session boş ise yenisini oluşturur ve Session'a ekler.

```
namespace GuvenliTasarimDesenleri.MvcWebUI.Controllers
{
   public class SepetKontrolor : Controller
    {
        //sepete ekleme işlemi yapılıyor
     private IUrunServis _urunServis;
     public SepetKontrolor(IUrunServis urunServis)
        {
            _urunServis = urunServis;
        }
 public RedirectToRouteResultResult SepeteEkle(int urunId)
        {
Urun urun = _urunServis.Get(urunId);
var sepet = (Sepet)Session["sepet"];
// eğer böyle bir sepet yoksa yeni bir sepet oluşturulur session ona eklenir.
            if (sepet == null)
            {
                sepet = new Sepet();
                Session["sepet"] = sepet;
            }
            // daha sonra ürünün sepete eklemesi işlemi gerçekleştirilir.
            sepet.SepeteEkle(urun, 1);
            return RedirectToAction("Index",sepet);
        }
```

Şekil 4.10 Güvenli Tasarım Deseni Konseptinde Sepete Ürün Ekleme İşlemi

```
1  public ViewResult Index()
2          {
3              var sepet = (Sepet) Session["sepet"];
4              return View(sepet);
5          }
6        // sepetten ürün çıkarma aksiyonu
7  public RedirectToRouteResultResult UrunCikar(int urunId)
8          {
9              Urun urun = _urunServis.Get(urunId);
10            var urun = (Urun)Session["urun"];
11             // sepetten ürün çıkarılıyor
12             urun.UrunCikar(urun);
13             return RedirectToAction("Index",sepet);
14         }
15 public ActionResult Checkout()
16         {
17             return View();
18         }
19     }
20 }
21
```

Şekil 4.11 Güvenli Tasarım Deseni Konseptinde Sepete Ürün Ekleme İşlemi

Sepet aksiyonları tüm e-ticaret sitelerinde bulunur. Geliştirme ortamına bağlı bir süreç değildir evrenseldir. Bu yüzden sepet ile ilgili işlemler geliştirme ortamına bağıl konumlandırılmak yerine örnekte verildiği gibi güvenli tasarım desenlerine uygun biçimde en tepeye yani varlıklar (entities) içerisine konulmalıdır. Android veya MVC uygulamaları bir e-ticaret sitesi için aynı sepete bağlanacağından sepette tutulacak bilgilere ait kodlar Varlıklar içerisindeki Sepet sınıfında yazılır aksiyonu ise kontrolörler içindeki SepetContoller içerisinde yapılır.

Güvenli tasarım desenleri ile örneklendirilecek bir diğer işlem ise gereklilik kontrollerinin sağlanmasıdır. Programlama dillerinde çeşitli onaylama kontrolleri (required validation) yer almaktadır. Örnek uygulamada sipariş verme aksiyonu gerçekleşirken sipariş formunda kullanıcı tarafından doldurulan alanların bazılarının zorunlu alan olması gerekebilir örneğin asp.net ile geliştirme yaparken araç kutusundan uygun bir tane zorunlu alan onaylama kontrolü sayfaya sürükle bırak ile eklenebilir. Bu durumda onaylama kontrolleri sayfanın ara yüz tarafına (front end) yerleştirilmiş olur fakat güvenli tasarım desenlerinin uygulanması söz konusu olduğundan .net framework altındaki Veri Notları (Data Annotations) kütüphanesinde yer alan geçerlilik kontrolleri güvenli tasarım desenlerine uygun bir biçimde öznitelik olarak kullanılır.

Kullanılan güvenli tasarım deseni yapısı ve diğer kodlamalar şekil 4.12'de gerçekleştirilmiştir.

```
namespace GuvenliTasarimDesenleri.Entities
{
    public class SiparisDetaylari
    {
        [Required(ErrorMessage = "İsim Alanı Zorunlu Alandır")]
        [Display(Name = "Ad Soyad")]
        public string Ad { get; set; }
        [Required()]
        [MinLength(10)]
        [MaxLength(50)]
        public string Adres { get; set; }
        [Required()]
        public string Sehir { get; set; }
    }
}
```

Şekil 4.12 Güvenli Tasarım Desenleri Kullanarak Gereklilik Kontrollerinin Uygulanması

Şekil 4.13 Doğrulama Kontrollerinin Uygulandığı Ekran Görüntüsü

Örnek uygulamanın ürün satışı ve sepet işlemlerine dair uygulanması gereken güvenli tasarım desenleri genel anlamda bu şekilde süreçlere dahil olmaktadır. Sayfalara yetkisiz erişimlerin engellenmesi ve kontrol altına alınması için ise örnek uygulamanın yönetim paneline her kullanıcının ekleme, silme, güncelleme, düzenleme gibi işlemleri yapması engellenmelidir.

Oturum yönetimi ile ve roller bir tür kısıtlama konulabilir fakat güvenli tasarım deseni olarak kabul edilen güvenlik mekanizması MVC üzerinde bu işi çok daha kolay bir şekilde kontrol altına almaktadır. Çeşitli atakları önlemek ve yetkisiz erişimleri yasaklamak için kullanılan en temel öz nitelik kodu şekil 4.14' de gösterilmiştir.

```
1  namespace GuvenliTasarimDesenleri.MvcWebUI.Controllers
2  {
3      public class YonetimController : Controller
4      {
5          private IUrunServis _UrunServis;
6          public YonetimController(IUrunServis urunService)
7          {
8              _UrunServis = urunServis;
9          }
10          public ActionResult Index()
11          {
12              return View(_UrunServis.GetAll());
13          }
14 // bu annotation ile yani attribute ile YeniOlustur metodu güvenli hale getirilmektedir.
15          [Authorize]
16          public ActionResult Yeni()
17          {
18              return View(new Urun());
19          }
20          [HttpPost]
21          public ActionResult Yeni(Urun product)
22          {
23             return View();
24          }
25      }
```

Şekil 4.14 Yetkisiz Erişimleri Yasaklamak İçin Kullanılan Güvenli Tasarım Deseni

Güvenli oturum tasarım desenine örnek gösterebileceğimiz kodlar şekil 4.15 ’de gösterilmiştir.

```
[HttpPost]
        public ActionResult Index(string kullaniciAd, string kullaniciSifre)
        {
            //Session.Add("Kullanici", kullaniciAd);
            //ya da
            Session["Kullanici"] = kullaniciAd;
            return View();
        }

```

Şekil 4.15 Güvenli Oturum Tasarım Deseninin Uygulanması

Şekil 4.16 Güvenli Oturum Deseni Giriş Ekran Görüntüsü

Rol tabanlı erişim kontrolü tasarım desenine örnek gösterebileceğimiz kodlar şekil 4.17'de gösterilmiştir.

```
[Authorize(Roles = "Yonetici, YoneticiKullanici")]
        public class ControlPanelController : Controller
        {
            public ActionResult SetTime()
            {

            }
            [Authorize(Roles = "Yonetici")]
            public ActionResult ShutDown()
            {

            }
         }
public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();

            services.AddAuthorization(options =>
            {
                options.AddPolicy("RequireAdministratorRole",
                    policy => policy.RequireRole("Yonetici"));
            });
        }
```

Şekil 4.17 Rol Tabanlı Erişim Tasarım Deseninin Uygulanması

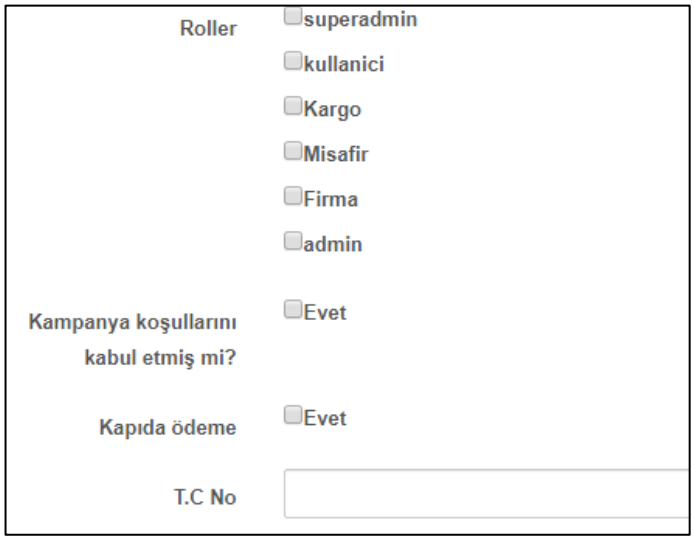
Sınırlı görünüm tasarım desenine örnek gösterebileceğimiz kodlar aşağıdaki gibidir. Sayfaların belirli rollerdeki kişilere gösterilmesini veya gizlenmesini kod tarafından hem oturum yönetimi ile hem de sayfa kriteri olarak uygulayan bir senaryoda ise kodlamanın nasıl olacağı şekil 4.18 ve şekil 4.19'te verilmiştir.

```
string page = Request.Url.LocalPath.Replace("/", "");

List<string> YoneticiPages = new List<string>()

{ "kullanici", "Basvurular ", "GelenTalepler ", "urunler ", "musteriler", "adresler ", "kargolar"};

List<string> BayiPages = new List<string>()

{ "urunlerim", "musterilerim", "adreslerim", "sattiklarim", "aldiklarim", "duyurularim"};

List<string> UyeKullaniciPages = new List<string>()

{ "aldiklarim", "sepetim", "faturalarim", "urunler", "iadeler", "kampanyalar", "siparislerim","adreslerim"};

List<string> NormalKullaniciPages = new List<string>()

{ "urunler", "kampanyalar", "dukkanlar", "indirimdekiler", "uyeol", "girisyap" };
```

Şekil 4.18 Sınırlı Görünüm Tasarım Deseninin Uygulanması

```
if (Session["Tur"] != " Yonetici ")
{
    if (YoneticiPages.Contains(page))
    {
        if (Session["Tur"] != "Yonetici")
        {
            Response.Redirect("DefaultController ");
        }
    }
    else if (BayiPages.Contains(page))
    {
        if (Session["Tur"] != "Bayi")
        {
            Response.Redirect("DefaultController");
        }
    }
    else if (UyeKullaniciPages.Contains(page))
    {
        if (Session["Tur"] != "Uye")
        {
            Response.Redirect("DefaultController");
        }
    }
    else if (NormalKullaniciPages.Contains(page))
    {
        if (Session["Tur"] != "Yeni")
        {
            Response.Redirect("DefaultController");
        }
    }
```

Şekil 4.19 Sınırlı Görünüm Tasarım Deseni Role Göre Görünüm



Şekil 4.20 Roller ve Rol Tabanlı Erişim Tasarım Deseni Uygulaması

Bilgi gizliliği tasarım desenine örnek gösterilecek kodlar şekil 4.21'daki gibidir. Bir web servis vasıtasıyla kullanıcıya ortak giriş olanağı sağlayan bir senaryoda web servisteki parola alanının MD5 işlemine tabii tutulduğu ve kullanıldığı kısımda bu kriptonun çözüldüğü bir işleyişin uygulaması şekil 4.21'da gösterilmiştir.

```
var yonetim =
a.yoneticiSifreKontrol("ornek_web_servis_KullaniciAdi", "ornek_web_servis_sifre",
txtKulAdi.Text, CalculateMD5Hash(txtParola.Text).ToUpper(), "");

                if (yonetim.FirstOrDefault().Sucess == true)
                {
var yonetimLogin =  a.YonetimPersonelBilgiGetir("ornek_web_servis_KullaniciAdi ",
" ornek_web_servis_sifre ", txtKulAdi.Text, "");
                    Session["Parola"] = txtParola.Text;
                    Session["Tur"] = "Yonetim";
                    Session["Ad"] = txtKulAdi.Text;
yonetimLogin.FirstOrDefault().yoneticiPersonel.FirstOrDefault();
                    string ses= Session["Tur"].ToString();
                    Response.Redirect("Default.aspx");
                }
public static string MD5HashCozumle(string input)
        {
            MD5 md5 = System.Security.Cryptography.MD5.Create();
            byte[] inputBytes = System.Text.Encoding.ASCII.GetBytes(input);
            byte[] hash = md5.ComputeHash(inputBytes);
            // step 2, convert byte array to hex string
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < hash.Length; i++)
            {
                sb.Append(hash[i].ToString("X2"));
            }
               return sb.ToString();
        }
```

Şekil 4.21 Bilgi Gizliliği Tasarım Deseninin Uygulanışı

Kriptolama yöntemleri ile şifrelenen uygulama kimliği, kullanıcı kimliği ve parolanın veri tabanlarına ne şekilde kayıt edildiğini gösteren örnek veri tabanı görüntüsü şekil 4.22'de gösterildiği gibidir. Bu bilgiler ilgili alanlarda kullanılmak üzere veri tabanından çağrıldıklarında gerekli güvenli tasarım deseni ve fonksiyonlar şifrelemeyi çözecek ve akış devam edecektir.

ApplicationId	UserId	Password
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	7324636D-C05E-4BAA-BA0E-908F5AF38987	AX4DwYG3uiJCcv07HPiEYwpYuPI=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	BFC09554-1B8B-466D-8D4F-B043C937B3A7	CCIKLyX6AGm+2wxCt/PQFJtMZIE=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	4AB50C36-5014-4F35-889B-6562CCE62ECF	hd/DKEjlsA5ZAOjCMlbqOclZhek=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	23DFD0E5-79DF-44A3-A505-977D07A28C21	2NZu2RKrwLf4xANZHnFAnLCAZYc=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	537AD90D-B34B-4AE9-9342-88357A9B7ED3	WZ9Hrbw7gTOpzU6OJKEgjeq4Jjs=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	807166B9-0FC0-4194-81E0-8DE1B58364DA	MKDUb6sfpxPcAGxO0ug8+iQpPrA=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	B8D884A5-251F-45F9-B3A4-5BAFAB1B2A47	ardPVKn6yKyUe+FE+rMvA5yaD/s=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	48249222-20F9-4E1F-AE00-B86B9951AA68	u+U/BYnvOpTF2UbgYw7YOWkj7y4=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	1472D2EB-98C6-49A4-8D7E-EFC02F4B4D2A	K6mRcI8heO8+LOwKmgmnOx3NtX4=
B50EB567-80D0-46FD-85E0-BEF0EFAA2147	FAFF56F6-6869-41C0-B270-741E2A6D5F37	X0B7zNoa09LYjHk9fS/avDC5R1E=

Şekil 4.22 Bilgi Gizliliği Tasarım Deseni Veri Tabanı

Güvenli dizin tasarım desenine örnek gösterilebilecek kodlar şekil 4.23'de verilmiştir. MVC içerisindeki kütüphanelerin yardımı ile [ServiceContract] veya [OperationContract] deyimleri temel bir işlemde yeterli olmaktadır.

```
1  [ServiceContract]
2    public interface IUrunServis
3    {
4        [OperationContract]
5        List<Urun> GetAll();
6        [OperationContract]
7        Urun Get(int productId);
8        [OperationContract]
9        void Add(Urun product);
10       [OperationContract]
11       void Update(Urun product);
12       [OperationContract]
13       void Delete(int productId);
14   }
```

Şekil 4.23 Güvenli Dizin Tasarım Deseninin Uygulaması

5. SONUÇ

Bu çalışmada tasarım desenlerinin yazılım geliştirmede sık karşılaşılan sorunlara yeniden kullanılabilir çözümler getirdiği, güvenlik problemlerine karşı ise kullanım seviyelerine göre farklı tasarım desenlerinin bulunduğuna dair konulara yer verilmiştir. Çalışma sonucunda güvenli tasarım desenlerinin, çeşitli güvenlik mekanizmalarını tanımlamak yerine, yazılım güvenlik açıklarına karşı güvenli geliştirme sürecini tanımladığı ve rehberlik yaptığı söylenebilir.

Tezde açıklanan güvenli tasarım desenlerinden bazıları örnek bir proje içerisinde kullanılmıştır. Ayrıca örnek proje içerisinde güvenli tasarım desenlerinden bir ya da birkaçının olduğu gibi uygulanması yerine nesne tabanlı programlama yaklaşımında sınıfların oluşturulma şekli ve aksiyonların kullanım tarzı ile de aslında güvenli tasarım desenleri uygulanmıştır. Bu şu anlama geliyor ki bir yazılım projesinde herhangi bir güvenli tasarım desenini illa kodlama ile kullanmak yerine nesne tabanlı programlama yaklaşımının sunmuş olduğu esnek yapıyı güvenli tasarım deseni düşüncesi ile yoğurarak geliştirme tarzına da güvenli tasarım desenleri yaklaşımları katılmaktadır.

Sonuç olarak güvenli tasarım desenlerin yazılım güvenlik açıklarına karşı mimari, tasarım ve uygulama düzeyinde incelenmiş ve geliştirilen örnek uygulama üzerinde belli başlı güvenli tasarım desenleri uygulanmıştır. Bu çalışmada açıklanan güvenli tasarım desenleri yazılım güvenlik açıklarına karşı oluşturulmuş iyi bir koleksiyon temsil etmektedir. Bu nedenle açıklanan güvenli tasarım desenlerinin kullanım şekli araştırmacının veya yazılım geliştiricinin çalışmakta olduğu projeye, platforma ve hedefine bağlı olarak farklılık gösterebilir.

Güvenli tasarım desenlerinin kullanıldığı projelerde genel olarak aşağıdaki faydalı sonuçlar ile karşılaşılmaktadır. Tablo 5.1’ de örnek projede uygulanan güvenli tasarım desenlerinden bazılarına ilişkin sonuçlar verilmiştir. Bu tablo sayesinde ilgili güvenli tasarım deseninin kullanılması ile ne gibi avantajlar elde edileceği anlaşılmaktadır.

Güvenli Tasarım Deseni Adı	Sonuç
Güvenli Kaydedici	Sistem arızaları ve diğer tüm aksiyonlar günlüğe kaydedilmiş olur ve problem durumunda uyarı verir.
Giriş Doğrulama	Merkezi kullanıcı havuzu yöntemi sayesinde kullanıcı ve veri işlemleri verimli bir şekilde gerçekleştirilmiş olur.
Çok Düzeyli Güvenlik	Güvenlik önlemlerinden bir veya birden fazlasının çalışmaması durumunda veya yanlış yapılandırılması durumunda gelebilecek saldırılara karşı daha az zarara uğrama ya da daha az saldırıya maruz kalması sağlanır.
Güvenli Kanallar	Sahte verilerin işleme alınması veya proje içerisinde yayılması riskini en aza indirir.
Güvenli Dizin Kontrollü Nesne Fabrikası	Temel işlevler dışındaki tüm işlemlerin başlangıç esnasında çalışması önlenmiş olur.
Güvenli Oturum Kimlik Doğrulayıcı Yetkilendirme Rol Tabanlı Erişim	Birden fazla hatalı giriş denemesi durumunda sistemin kilitlenmesi sağlanır.
Kimlik Doğrulayıcı Kontrol Noktası	Süresi dolan veya kullanılmayan tüm kullanıcı hesaplarının kaldırılması veya izinlerinin devre dışı kalması sağlanır.
Güvenli Kaydedici Güvenli Durum Makinesi	Proje içerisinde çalışan tüm uygulama ve ağ etkinliklerinin günlüklere yazılması ve takip edilmesi sağlanmış olur.
Güvenli Strateji Fabrikası Güvenli Fabrika	Projeler içerisindeki güvenlik mimarisinin çok daha karmaşık bir hale gelmesi sağlanır.
Ayrıcalık Ayrıştırma Hatalarla Tam Görünüm	Proje içerisinde yer alan uygulamaların herhangi birinde meydana gelebilecek arıza durumunda hizmetin tamamının durdurulmasına neden olmadan çalışır.
Hatalarla Tam Görünüm Sınırlı Görünüm	Proje içerisinde herhangi bir modülün kodlanması esnasında yapılabilecek bir hataya karşın tüm kaynak kodların kullanıcıya veya zararlı kişilere gösterilmesi engellenmiş olur.

Tablo 5.1 Güvenli Tasarım Desenlerinin Projede Kullanılmasına İlişkin Sonuçlar

Tablo 5.2' de güvenli tasarım desenlerinin Microsoft .net kütüphanesi ile geliştirilen yazılımlarda karşılık geldiği yapılar, metotlar veya fonksiyonlar gösterilmiştir. Özellikle Microsoft Asp.net MVC' nin de bir tasarım deseni olduğu düşünüldüğünde .net kütüphanesi içerisinde pek çok tasarım deseni ve güvenli tasarım desenine çekirdek yazılımda rastlamak mümkün olmaktadır. Microsoft .net geliştirme ortamında üyelik işlemleri ve rol tabanlı işlemleri gerçekleştirirken geliştiricinin kendisine özgü kriptolama yöntemleri olacağı gibi bu tür bir geliştirmede roller güvenli tasarım deseni veya kimlik doğrulayıcı tasarım deseni gibi örnek alınacak tasarım desenlerinden faydalanılmaktadır. Ayrıca .net içerisinde zaten hali hazırda yer alan bir takım sınıf veya metot ile de güvenli tasarım deseni standardına uygun geliştirme yapılabilir. Aşağıda bulunan tablo 5.2 bunun gibi zaten .net kütüphanesinde yer alan güvenli tasarım desenlerine destek niteliğindeki sınıfların, metotların veya yapıların karşılık geldiği güvenli tasarım desenlerini göstermektedir.

Güvenli Tasarım Deseni Adı	**İşlev**
Güvenli Fabrika Tasarım Deseninin Projede Karşılık Geldiği .Net Yapı ve Metotları	IDbCommand.CreateParameter Convert.ToBoolean WebRequest: HttpWebRequest Asp.Net Runtime
Güvenli Strateji Fabrikası Tasarım Deseninin Projede Karşılık Geldiği .Net Yapı ve Metotları	Site Map Providers Session State Providers Web Event Providers Web Part Personalization Providers
Rol Tabanlı Erişim, Roller, Güvenli Dizin ve Kimlik Doğrulayıcı Tasarım Desenlerinin Projede Karşılık Geldiği .Net Yapı ve Metotları	Role Providers Profile Providers Membership Providers

Tablo 5.2 Güvenli Tasarım Desenlerinin Projede Karşılık Geldiği .Net Yapı ve Metotları

Tablo 5.3' de güvenli tasarım desenlerinin kullanımının genel olarak geliştiricilere ve projelere sağladığı katkılara ilişkin sonuçlara değinilmiştir. Yazılım geliştirme yaşam döngüsünde tasarım desenlerinin ve güvenli tasarım desenlerinin kullanımına ilişkin olumlu sonuçlar elbette tabloda yer alan maddelerle sınırlı değildir. Geliştiricinin izleyeceği yöntem ve metotlara göre güvenli tasarım desenlerinin kullanımından daha fazla olumlu sonuç elde etmek mümkündür.

Güvenli Tasarım Deseni Adı	**Sonuç**
Güvenli Sorumluluk Güvenli Fabrika	Projeler içerisinde çalışan tüm metotlar, fonksiyonlar veya servisler kabul edilebilir bir korunma seviyesi ile çalışırlar [25].
Çok Düzeyli Güvenlik Güvenli Strateji Fabrikası	Projelerde yer alan uygulamalar önemsiz güvenlik açıklarına maruz bırakılmadan geliştirilmiş olurlar.
Güvenli Oluşturucu Fabrikası Güvenli Sorumluluk Zinciri	Sorun giderme, test etme, denetim izleri etkin bir şekilde kurgulanır.
Güvenli Fabrika Güvenli Oluşturucu	Standart güvenlik tasarım desenlerinin kullanılması ortak bir geliştirme teknolojisi ve standart sağlar.
Ayrıcalık Ayrıştırma Güvenli Kaydedici Güvenli Sorumluluk Zinciri	Büyük ölçüde geliştirilmiş güvenlik ilkesi bilincinin benimsenmesini sağlar.
Kontrollü Nesne Fabrikası Güvenli Fabrika Çekirdeğe Erteleme	Veri bütünlüğü korunmuş olur.
Güvenli Fabrika Güvenli Strateji Güvenilmez Ayrışma Hassas Bilgileri Temizle	Tasarım desenleri test edilmiş ve kanıtlanmış oldukları için gelişim paradigmaları sağlayarak geliştirme sürecini hızlandırırlar.
Güvenli Strateji Güvenilmez Ayrışma Çekirdeğe Erteleme Kontrollü Nesne Fabrikası	Güvenli tasarım desenlerinin kullanılması, büyük sorunlara neden olabilecek küçük sorunları önlemeye yardımcı olur ve bu tür desen kalıplarına aşina olan geliştiriciler için kod okunabilirliğini arttırır.

Tablo 5.3 Güvenli Tasarım Desenlerinin Genel Olarak Kullanılmasına İlişkin Sonuçlar

Bu çalışma ile güvenli tasarım desenlerinin yazılım güvenlik açıklarına karşı incelenmesi ve belli oranda uygulanması sağlanarak yazılım geliştiriciler için veya mimari seviyede algoritma geliştirenler için güvenli tasarım desenlerinin önemine yer verilerek farkındalık kazandırılmıştır. Güvenli tasarım desenleri ile ilgili yapılan farklı çalışmalarda yalnızca bir konu üzerine durulmuş olduğundan literatürde adı geçen ve kendini kabul ettiren güvenli tasarım desenlerinin hepsine bu çalışma içerisinde yer verilmiştir. Bu çalışmayı inceleyecek olan araştırmacılar soyut bir kavram olan güvenli tasarım desenlerinin kod ile nasıl entegre edildiğini somut bir şekilde görebileceklerdir. Soyut kavramların örneklendirildiği, kod ve ekran görüntüleri ile desteklendiği ve faydalar tablosunda ne işe yaradığı ile ilgili verilen tüm bilgiler diğer çalışmalara nazaran soyut kavramların somutlaştırılması sebebiyle literatürdeki diğer çalışmalardan farklıdır. Konuları tanım olarak açıklama yerine örneklendirilerek pekiştirilmesi sebebiyle hem tez çalışmasında literatürde yenilik değeri bulunmakta hem de ileride bu konu üzerine çalışma yapacak olan araştırmacılara temel niteliğinde rehberlik sağlayacaktır. Yapılan çalışma güvenli tasarım desenlerinden literatürde bilinen ve bilinmesi gerekenleri tek bir yerde derleyip Türkçe olarak hazırlanması sebebiyle de alanında yeni bir çalışma olmuştur. Ayrıca güvenlik ile ilgili sivil kuruluşların veya devlet kurumlarının yapmış oldukları çalışmalarda genellikle geliştirme sonrasında alınacak önlemlere yer verildiğinden bu tez çalışmasının geliştirme aşamasında alınacak olan önlemlere yer veriyor olması sebebiyle yeni bir bakış açısı kattığı söylenebilir. Güvenli tasarım desenleri konusu çok geniş ve eski bir tarihe dayanan konu başlığı olmasına rağmen, soyut seviyede olması sebebiyle araştırmacıların veya yazılım geliştiricilerin üzerine çok fazla eğilmediği bir konudur. Bu çalışma güvenlik zafiyetlerinin sonradan ortadan kaldırılması yerine önceden önlemlerinin alınması, nizami bir prensip ile kod yazılması, kod okur yazarlığının arttırılması, karmaşık programlamanın önüne geçilmesi ve katmanlı mimaride çalışılması gibi metotların etkinliğinin arttırılması için ortaya somut bir yöntem koymaktadır. Bu çalışma ileride literatüre yeni kazandırılmış olan güvenli tasarım desenleri ile genişletilerek örnek kodlama ile pekiştirilmesi sağlanarak geliştirilebilir.

KAYNAKLAR

[1] Kent B., ve Ward C., Nesne Yönelimli Programlar İçin Tasarım Desenlerini Kullanma, Nesneye Dayalı Programlama, Sistemler, Diller ve Uygulamalar Konferansı, 1987.

[2] Gamma, Erich, Helm, Richard, Johnson, Ralph, ve ark., Tasarım Desenleri, Nesne Yönelimli Yazılımın Yeniden Kullanılabilir Elemanları, 1995.

[3] Grady Booch, Yazılım Arkeolojisi ve Yazılım Mimarisinin El Kitabı, Yazılım Yeniden Yapılandırma Çalıştayı, LNI Dergisi sayı 126 sayfa 5, 2008.

[4] Mario Barbacci ve ark., Yazılım Kalite Özellikleri Teknik Raporu, Yazılım Mühendisliği Enstitüsü, Adres: hhttp: //www.sei.cmu.edu/reports/95tr021.pdf, 2005.

[5] L. Chung, B.A. Nixon, ve E. Yu. Yazılım Mühendisliğinde Fonksiyonel Olmayan Gereksinimler, Adres: http://books.google.com/books?id=IgV_nRpf5tUC., 2000.

[6] Joseph Yoder ve Jerey Barcalow, Uygulama Güvenliğini Sağlamak İçin Mimari Desenler, Programların Örüntü Dilleri Dördüncü Konferansı, 1998.

[7] L. Chung, B.A. Nixon, ve E. Yu, Yazılım Mühendisliğinde Fonksiyonel Olmayan Gereklilikler Kitabı, adres: http://books.google.com/books?id=IgV_nRpf5tUC. 2000.

[8] Y.Dangler J, Güvenli Tasarım Desenlerinin Kategorilendirilmesi, Doğu Tennese Devlet Üniversitesi, 2013.

[9] Tubitak Bilgem, Güvenli Yazılım Geliştirme Temel Kuralları Dokümanı, Tubitak Bilgem Siber Güvenlik Enstitüsü, 2014.

[10] Gündüz S. ve Aydoğdu D., Web Uygulama Güvenliği Açıkları ve Güvenlik Çözümleri Üzerine Bir Araştırma, 2016.

[11] Wang J., Eong S. ve Chung L., Güvenli Tasarım Desenlerinin Analiz Edilmesi, Örnek Konu e-Ticaret Sistemleri, 2014.

[12] Fernandez E., ve Petrie M., Güvenli Desenler ve Güvenli Sistemlerin UML Diyagramları ile Birlikte İncelenmesi, 2007.

[13] Eduardo B., ve Fernandez E., Güvenlik Modellerini Kullanarak Güvenli SCADA Sistemleri Tasarlama, Hawaii Uluslararası Sistem Bilimleri Konferansı, 2010.

[14] Steve R., ve ark., Güvenli Sistemler Kurmak İçin Güvenli Desenler Oluşturmak, Avrupa Güvenilir Bilgi İşlem Konferansı, 2015.

[15] Eduardo B., ve Fernandez E., Güvenli Tasarım Desenleri Kullanılarak Güvenli Bulut Mimarileri Oluşturma, Uluslararası Bulut Mühendisliği Çalıştayı Konferansı, 2016.

[16] Obaid U., Natasa Z., Akıllı Ölçüm Sistemlerinde Güvenlik İçin Güvenli Tasarım Desenleri, IEEE Avrupa Modelleme Sempozyumu, 2015.

[17] Chad D., Kirk S., Robert C., Güvenli Tasarım Desenleri, Yazılım Mühendisliği Enstitüsü, Mart 2009, Güncelleme Ekim 2009.

[18] Schumacher M., Güvenlik ve Sistem Mühendisliğini Bütünleştirmek, Yazılım Tasarım Desenlerinde Güvenlik Kalıpları Wiley Serisi, Adres: http://books.google.com/books?id=gtpQAAAAMAAJ. 2006.

[19] Alexander C., Ishikawa S., Silverstein M., Bir Desen Dili, Oxford Üniversitesi Yayınları, Adres: http://books.google.com/books?id=hwAHmktpk5IC., 1977.

[20] Dangler Y., Güvenli Tasarım Desenlerinin Sınıflandırılması, Doğu Tennese Devlet Üniversitesi, 2013.

[21] CERT Güvenli Kodlama Standartları,
Adres: http://www.securecoding.cert.org/confluence/x/BgE. 2009.

[22] Robert C., CERT Güvenli Kodlama Standardı, 2008.

[23] Schumacher M., Güvenlik ve Sistem Mühendisliğini Bütünleştirmek, Yazılım Tasarım Desenlerinde Güvenlik Kalıpları Wiley Serisi, adres: http://books.google.com/books?id=gtpQAAAAMAAJ. 2006.

[24] Chad D., Kirk S., Robert C., Yazılım Mühendisliği Enstitüsü Teknik Rapor,
Adres: http://www.sei.cmu.edu/reports/09tr010.pdf. 2009.

[25] Sasha R., Güvenli Tasarım Desenleri Bölüm 1, v1.4, 2001.

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı	: Fatih KAPLAN
Doğum Yeri ve Tarihi	: Selçuklu 1993

Eğitim Durumu

Lisans Öğrenimi	: KTO Karatay Üniversitesi Bilgisayar Mühendisliği
Yüksek Lisans Öğrenimi	: Elektrik Bilgisayar Mühendisliği Tezli Y.L. (Devam Ediyor)
Bildiği Yabancı Diller	: İngilizce
Bilimsel Faaliyetleri	: KODLAB yayınları asp.net ile proje geliştirme kitabı

İş Deneyimi

Stajlar	: Uzman IT, Akınsoft, Vodafone, Konya Büyükşehir Belediyesi
Projeler	: Asp.net ile e ticaret geliştirme, mini ebys yazılımı, operating system simulation, analysis of algorithms, Maden ocağı takip sistemi, sistem odası ısı-nem takip sistemi, dijital baskı reçete yazılımı, klinik yazılımı, SSO, Fazla mesai yazılımı, Kalite yönetim sistemi yazılımı, bilgi işlem iş takip yazılımı, öğrenci staj otomasyonu, tercih robotu,
Çalıştığı Kurumlar	: Uzman IT, Motifli Medya, Akınsoft, Vodafone Quala Tech., M.E.B Karatay Halk Eğitim Merkezi, PAUS Web Yazılım, KTO Karatay Üniversitesi

İletişim

E-Posta Adresi	: fatih.kaplan@karatay.edu.tr

Tarih : 02/10/2019