



BİLİŞİM VE BİLGİ GÜVENLİĞİ İLERİ TEKNOLOJİLER ARAŞTIRMA MERKEZİ

GÜVENLİ YAZILIM GELİŞTİRME KILAVUZU

Rev: 1.0

16 Nisan 2018

© 2018 TÜBİTAK BİLGEM
Bilişim ve Bilgi Güvenliği İleri Teknolojiler Araştırma Merkezi

P.K. 74, 41470 Gebze / KOCAELİ
Tel: (0262) 648 10 00, Faks: (0262) 648 11 00
www.bilgem.tubitak.gov.tr



Bu doküman, alıntı vererek kullanılabilir ya da paylaşılabilir ancak değiştirilemez ve ticari amaçla kullanılamaz. Detayları <https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode.tr> bağlantısından erişebilirsiniz.

İÇİNDEKİLER

ŐEKİLLER LİSTESİ	4
TABLOLAR LİSTESİ.....	4
TANIMLAR	5
KISALTMALAR.....	5
1. GİRİŐ	6
1.1. Özet	6
1.2. Amaç ve Kapsam	6
2. GÜVENLİ YAZILIM GELİŐTİRME	7
2.1. YAZILIM GÜVENLİĐİ İLKELERİ	7
2.1.1. Gerek Duyulan En Az Yetkiyi Ver	7
2.1.2. Tüm EriŐimleri Denetle	8
2.1.3. Yetkileri Ayr	9
2.1.4. Varsayılan Deđerleri Güvenli Hale Getir	9
2.1.5. Ortak EriŐilen Kaynaklara Farklı Kanallardan EriŐ	10
2.1.6. En Zayıf Halkayı Tespit Et ve Güçlendir	10
2.1.7. Saldırı Yüzey Alanını Azalt	10
2.1.8. Savunma DerinliĐi OluŐtur	10
2.1.9. Basit Güvenlik Mekanizması Tasarla.....	11
2.1.10. AnlaŐılabilir ve Kolay Kullanılabilir Güvenlik Mekanizması Tasarla	11
2.2. YAZILIM GÜVENLİĐİNİ SAĐLAMA YÖNTEMLERİ.....	12
2.2.1. Sezgisel (Heuristic) ve Benzetime (Simulation) Dayalı Yöntemler.....	12
2.2.2. Biçimsel (Formal) Yöntemler	14
2.2.3. Yazılım ÇalıŐma Ortamı GüvenliĐini Sađlayan Yöntemler.....	16
2.2.4. Birden Fazla Araçla Analiz Yöntemleri	17
2.2.5. Yazılım GüvenliĐini Sađlamada Diđer Yöntemler	18
2.3. GÜVENLİ YAZILIM GELİŐTİRME YAŐAM DÖNGÜŐÜ	19
1.1.1 GeliŐtiricilerin Güvenli Yazılım GeliŐtirme Konusunda EĐitilmesi.....	19
1.1.2 Yazılım Güvenlik Gereksinimlerinin Tanımlanması	20
1.1.3 Güvenlik Tasarımı ve Mimarisi	24
1.1.4 Güvenli Kodlama	26
1.1.5 Güvenli Kurulum	26
1.1.6 Güvenlik Analizleri ve Testleri.....	27
2.4. GÜVENLİK TEST ARAÇLARI	31
2.4.1. Kaynak Kod Analizi Test Araçları	31
2.4.2. Sızma Testi Araçları	32
2.4.3. Rastgele (Fuzz) Test Araçları	33
2.5. GÜVENLİ YAZILIM GELİŐTİRME OLGUNLUK MODELLERİ.....	35
2.5.1. SAMM.....	35

2.5.2. BSİMM.....	36
2.6. UYGULAMA GÜVENLİĐİ.....	37
KAYNAKÇA	40
EK 1: GÜVENLİ YAZILIM GELİŐTİRME DENETİM LİSTESİ	41
EK 2: UYGULAMA GÜVENLİĐİ KURALLARI	44
EK 2-Lahika-01: Mimari, Tasarım ve Tehdit Modelleme	53
EK 2-Lahika-02: Kimlik Doğrulama	55
EK 2-Lahika-03: Dosyalar ve Kaynakların GüvenliĐi	63
EK 2-Lahika-04: Oturum Yönetimi.....	66
EK 2-Lahika-05: EriŐim Denetimi	69
EK 2-Lahika-06: Güvenli Kurulum ve Yapılandırma	72
EK 2-Lahika-07: Güçlü Kriptografik Mekanizmaların Kullanımı	75
EK 2-Lahika-08: Veri Koruma.....	77
EK 2-Lahika-09: Hata Ele Alma ve Kayıt.....	79
EK 2-Lahika-10: İletişim GüvenliĐi	81
EK 2-Lahika-11: İş MantiĐı.....	85
EK 2-Lahika-12: Kötücül İşlemleri Engelleme	86
EK 2-Lahika-13: Mobil Uygulama GüvenliĐi	87
EK 2-Lahika-14: Girdi ve Çıktı Süzme.....	88
EK 2-Lahika-15: Web Servisleri GüvenliĐi	91
EK 2-Lahika-16: Kişisel Verilerin Korunması.....	93

ŐEKİLLER LİSTESİ

Őekil 1 İstek Alınırken ve Cevap Verilirken Yapılması Gereken Yetkilendirmeler	8
Őekil 2 Yetkilerin Ayrımı İin Örnek Koşullar.....	9
Őekil 3 Yazılım Güvenlik Gereksinimleri Tanımlama Adımları	20
Őekil 4 Yazılım Varlıkları Deęerlendirme Matrisi	21
Őekil 5 Tehdit Profili Deęerlendirme Matrisi	22
Őekil 6 SAMM Olgunluk Modeli Bileşenleri	35
Őekil 7 BSIMM Etkinlik Alanları ve Faaliyetleri.....	36
Őekil 8 Örnek bir firma için BSIMM deęerlendirmesi.....	37

TABLolar LİSTESİ

Tablo 1 Statik Analiz Araları.....	31
Tablo 2 Keşif ve Sızma Testi Araları.....	32
Tablo 3 Web Sızma Testi Araları	33
Tablo 3 Rastgele (Fuzz) Test Araları	34
Tablo 4 Uygulama Kapsamı ve Uygulama Güvenlięi Kapsamı.....	37
Tablo 5 Sektörlere Göre Uygulamaların Güvenlik Seviyelerinin Sınıflandırılması	39

TANIMLAR

Güvenli Yazılım	Olası saldırılara karşı önlemleri olan, saldırıyı önleyemese bile çalışma sürecini doğru bir şekilde devam ettirebilecek ve kötüye kullanıldığı durumları fark edebilecek yazılım mimarisine ve tasarımına sahip yazılım
Güvenli Yazılım Geliőtirme	Yazılım geliőtirme süreci aŐamalarının (gereksinim, tasarım, geliőtirme, test dođruma vb.) bütününi içeren Yazılım Geliőtirme YaŐam Döngüsü (YGYD) süresince Güvenli Yazılım geliőtirmek için yürütölen faaliyetlerin tümü
Risk	Belirsizliklerin amaçlara olan etkisi (ISO 31000)
Uygulama	Bir kuruluŐun kullanıcılarının, belirli görevleri yerine getirmek veya belirli türde bilgi teknolojisi problemlerini çözmek için bir iŐ süreci veya iŐlevini otomatik hale getiren, uygulama yazılımı, uygulama verisi ve ilgili süreçleri içeren bir bilgi teknolojisi çözümi
Yazılım Güvenliđi	Yazılımın, saldırı veya tehdit altındayken iŐlevlerini dođru bir şekilde yerine getirmeye devam edecek şekilde korunmasına
Yazılım Güvenlik Güvencesi	Bir yazılımın, iŐletim halindeyken isteyerek oluşturulmuŐ hatalara rađmen güvenilir şekilde çalıŐmaya devam edebileceđinin dođrulanabilirliđi

KISALTMALAR

ASVS	OWASP Application Security Verification Standard
BİLGEM	TÜBİTAK BiliŐim ve Bilgi Güvenliđi İleri Teknolojiler AraŐtırma Merkezi
BSIMM	Building Security In Maturity Model / Güvenlik Geliőtirme Olgunluk Modeli
CWE	Common Weakness Enumeration
GYGK	Güvenli Yazılım Geliőtirme Kılavuzu
YGYD	Güvenli Yazılım Geliőtirme YaŐam Döngüsü
OWASP	Open Web Application Security Project
SAMM	Software Assurance Maturity Model / Yazılım Güvencesi Olgunluk Modeli
TÜBİTAK	Türkiye Bilimsel ve Teknolojik AraŐtırma Kurumu
UGK	Uygulama Güvenliđi Kuralı
YGYD	Yazılım Geliőtirme YaŐam Döngüsü

1. GİRİŐ

1.1. Özet

Yazılımın, saldırı veya tehdit altındayken işlevlerini doğru bir şekilde yerine getirmeye devam edecek şekilde korunmasına, yazılım güvenliđi denir. Yazılım güvenliđi faaliyetlerinin amacı tüm bilgi güvenliđi saldırılarına karşı daha dirençli ve hatasız çalışan yazılım üretmektir.

Dokümanın ilk bölümünde dokümanın amacı ve kapsamı paylaşarak yazılıma yönelik tehditler hakkında bilgi verilmiştir. Sonraki bölümde yazılım güvenliđi ilkeleri ve yöntemleri anlatılarak güvenli yazılım geliştirme yaşam döngüsü ve yazılım güvenliđinin nasıl ölçülebileceđi ile ilgili bilgiler sunulmuştur. Anlatılan konuları destekleyecek önerilen güvenli yazılım geliştirme denetim listesi (EK-1) ve uygulama güvenliđi kuralları (EK-2), dokümanın sonunda ek olarak verilmiştir.

1.2. Amaç ve Kapsam

2016-2019 Ulusal Siber Güvenlik Stratejisi ve Eylem Planı eylemleri kapsamında, yazılım projelerinde yazılım geliştirici ve proje yöneticileri tarafından kılavuz olarak kullanılmak üzere TÜBİTAK-BİLGEM ve Ulaştırma Denizcilik ve Haberleşme Bakanlığı işbirliđi ile hazırlanan Güvenli Yazılım Geliştirme Temel Kuralları Dokümanının güncellenmesi planlanmıştır. Bu doğrultuda Kalkınma Bakanlığı işbirliđi ile “Siber Güvenlik Eğitim ve Araştırma Merkezi” projesi kapsamında aşağıdaki ihtiyaçların karşılanması faydalanılabilmesi için bu doküman hazırlanmıştır.

- Yazılım projelerinde görev alacak yazılım geliştiricilerin eğitimi
- Kurumsal güvenli yazılım geliştirme kültürünün oluşturulması
- Yazılım projeleri boyunca kullanılacak süreçleri tanımlarken güvenli yazılım geliştirme kurallarının nasıl ve ne zaman uygulanacağına belirlenmesi
- Yazılımların siber güvenlik tasarımının yapılması ve güvenlik tasarım dokümanının oluşturulması
- Yazılım tasarımının siber güvenlik gözü ile gözden geçirilmesi
- Yazılım güvenlik testlerinin gerçekleştirilmesi
- Güvenli yazılım geliştirme yaşam döngüsü faaliyetlerinin izlenmesi ve denetimi
- Yazılım güvenliđinin sürekliliğinin sağlanması

2. GÜVENLİ YAZILIM GELİŐTİRME

Yazılıma karşı yapılan saldırıların üç ana amacı bulunmaktadır:

- Yazılımın çalışmasını tamamen durdurmak ya da doğru bir şekilde çalışmasını engellemek,
- İçine zararlı kod dahil ederek veya yazılımın kodunu deęiőtirmek yoluyla, yazılımın çalışma akışını deęiőtirerek istemeyen ya da öngörülmeven bir amaca hizmet etmesini sağlamak,
- Yazılımın çalışmasındaki işlemler ve çevresiyle ilgili bilgi edinerek yazılımın zayıf noktalarını tespit etmek, çalışma ortamına sızmak ve istenilmeyen amaçları gerçekleőtirmek için aracı olarak kullanmak

Yazılıma yapılan saldırıların önlenmesi için öncelikle yazılımların, yazılım güvenlięi ilkelerine uygun olarak tasarlanması ve geliştirilmesi, yazılım güvenlięini sağlayacak yöntemlerin ve araçların kullanılması gerekmektedir. Bütün bu çalışmalar da bütüncül bakış açısı ile güvenli yazılım geliştirme yaşam döngüsü içinde gerçekleştirilmesi ve bu konudaki olgunluęın geliştirilmesi gerekmektedir. Bu bağlamda aőaęıdaki bölümlerde sırası ile yazılım güvenlięi ilkeleri, yazılım güvenlięini sağlayan yöntemler, güvenli yazılım geliştirme yaşam döngüsü, güvenlik test araçları açıklanmıştır. Bu bölümün en sonunda da uygulama güvenlięi tanımlanarak bu kapsamda yapılması gereken faaliyetler özetlenmiştir.

2.1. YAZILIM GÜVENLİęİ İLKELERİ

Yazılım güvenlięi ilkeleri güvenlikle ilgili tüm mekanizmaların tasarımında kullanmak gereklidir. Teknik hususların yanı sıra insan etkileşimiyle ilgili hususların da göz önüne alması önemlidir. Aőaęıda teknik ve insan etkileşimlerini de kapsayacak şekilde, güvenli yazılım geliştirme yaşam döngüsünde dikkate alınması gereken temel yazılım güvenlięi ilkeleri anlatılmıştır.

2.1.1. Gerek Duyulan En Az Yetkiyi Ver

En az yetki ilkesi, her bir yazılım prosedürünün kendi işini bitirmesi için gerekli en az hak kümesiyle çalıştırılmasıdır. Bir öğeye (metot, kullanıcı, yazılım vb.) belirli bir görevi tamamlaması gereken en az yetki kümesi sağlanmalıdır. En az yetki ilkesinin yararları arasında şunlar sayılabilir:

- Sistem kararlılığına olumlu etki: uygulamanın sistem üzerinde yapabileceđi deđişiklikler sınırlanır
- Sistem güvenliğine olumlu etki: bir uygulamadaki zayıflıkların sistemin geri kalanına sızmak için kullanılamaz
- Konuşlandırma kolaylığı: bir uygulamanın daha az yetki gerektirmesi karmaşık bir ortama konuşlandırılmasını kolaylaştırır.

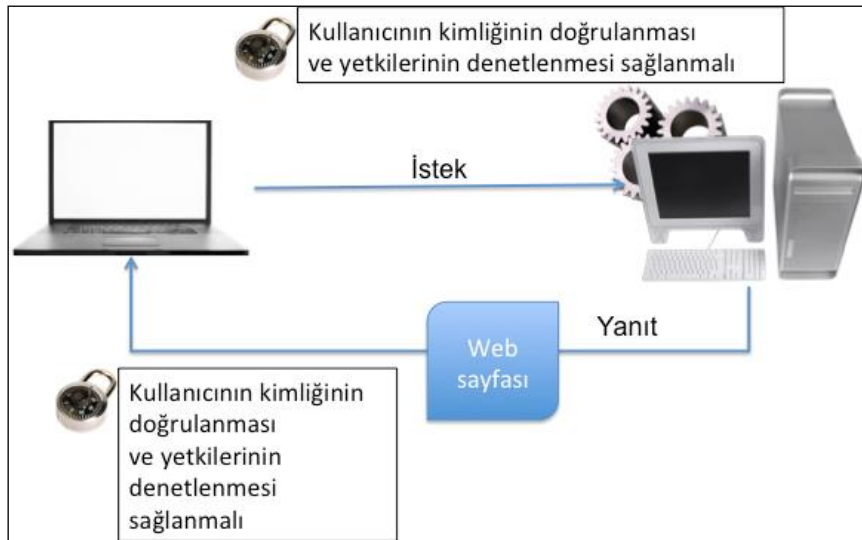
En az yetki ilkesi kapsamında ayrıca aŐağıdaki hususlara da dikkat edilmesi gerekmektedir.

- Sistem kaynaklarına erişmesi gereken uygulamalara yalnızca erişmesi gereken kısıtlı bölgeler için yetki verilmeli ve yönetici haklarıyla çalıştırılmamalıdır.
- Yükseltilmiş hakları gerektiren yazılım süreçleri bu haklara sadece gereken en az süre boyunca sahip olmalıdır.

2.1.2. Tüm Erişimleri Denetle

Her bir nesneye yapılan her bir erişimde yetki kontrolü yapılmalıdır. Bu ilkeye uyum, sistem genelinde erişim denetimi sağlanmasını gerektirir. Tüm erişimleri denetle ilkesi kapsamında ayrıca aŐağıdaki hususlara da dikkat edilmesi gerekmektedir.

- Normal durumların dışında, başlatma, geri kazanım, kapatma, bakım sayfalarında da erişim denetlenmelidir.
- Bir istek alındığında yapılan yetki kontrolü aynı isteđin yanıtı verilirken de yapılmalıdır (Őekil 1).

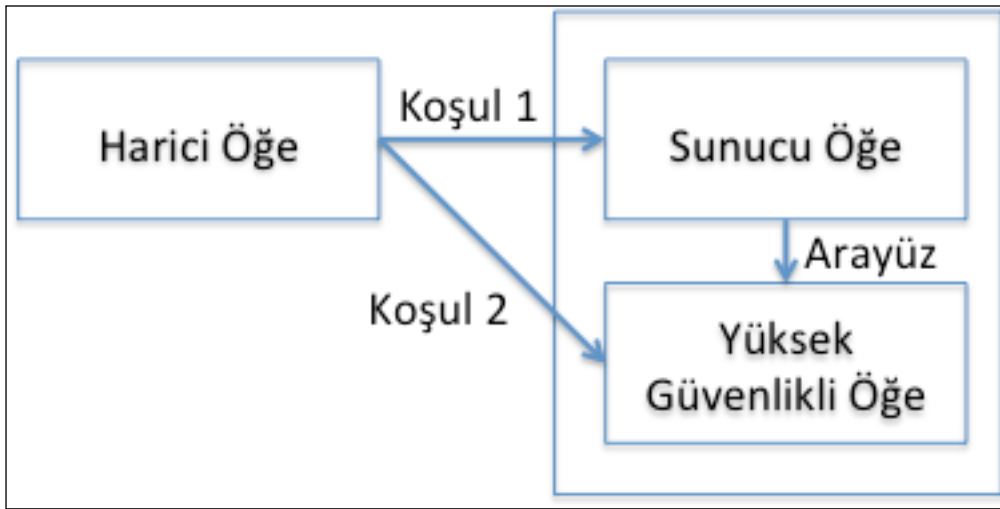


Őekil 1 İstek Alınırken ve Cevap Verilirken Yapılması Gereken Yetkilendirmeler

2.1.3. Yetkileri Ayır

Bir sistem yüksek güvenli bir işleme yalnızca tek bir koşula bađlı olarak izin vermemelidir. Yüksek güvenli bir işleme izin verilmesi için birden fazla koşulun sađlandığı dođrulanmalıdır. Örneđin: 75,000 TL üzeri çekler aynı şirkette iki yetkili tarafından imzalı olmalıdır.

Őekil 2’de yetkilerin ayrımı için bir örnek verilmiŐtir. Normal bir işlem için önce Sunucu Öđe’ye oturum açılmalıdır (KoŐul 1). Sonrasında yüksek güvenli işlem gerçekleŐtirmek için Yüksek Güvenlikli Öđe’ye oturum açılmalıdır (KoŐul 2). Yüksek güvenli işlem her iki koşul da gerçekleŐmeden koŐturulmamalıdır.



Őekil 2 Yetkilerin Ayrımı İçin Örnek KoŐullar

2.1.4. Varsayılan Deđerleri Güvenli Hale Getir

Kullanılan tüm varsayılan deđerlerin güvenliđi artıracak őekilde sečilmesi gerekmektedir. Bu konuda sık görülen hatalar őunlardır:

- Çalıştırılabilir dosyaların varsayılan olarak herkesin yazabileceđi őekilde kurulması
- Uygulama ana dizininin herkesin okuyabileceđi őekilde kurulması
- Herkesin yazabileceđi iz kaydı dosyaları
- Herkesin okuyabileceđi dosyalarda varsayılan (geliŐtirme, test) parolaların bulunması
- Herkes tarafından okunabilen dizinler
- Cihazlarda IP sahtekarlıđına izin veren varsayılan ayarlar kullanılması
- Paylaşılan anahtar dosya / veritabanlarında güvensiz hakların verilmiŐ olması

Bu gibi açıklıklara önlem olarak tüm varsayılan değerlerin gözden geçirilmesi ve güvenli değerlerin atanması gereklidir.

Varsayılan değerleri güvenli hale getirilmesi kapsamında ayrıca aşağıdaki hususlara da dikkat edilmesi gerekmektedir.

- Bir öğeye özellikle bir nesne üzerinde erişim verilmemiş ise bu öğeye erişimi olmamalıdır.
- Bir nesnenin varsayılan erişimi *hiç* olmalıdır.
- Parola süresinin dolması, karmaşıklık kontrolü gibi kontroller varsayılan durum olmalıdır.
- Kriptografik algoritmaların çalışma kümesi varsayılan olarak en güvenli küme olmalıdır.
- Kimlik denetiminde en güvenli faktör varsayılan olarak kullanılmalıdır.

2.1.5. Ortak Erişilen Kaynaklara Farklı Kanallardan Eriş

Kaynaklara erişen mekanizmalar ortak kullanılmamalıdır (örneğin paylaşılan dosyalar). Ortak kanallardaki bilgi akışından kaynaklanan bilgi sızması/bozulması problemleri ve yan kanal saldırıları gibi problemlere neden olmaktadır. Ayrıca sistem ve yazılım içerisinde farklı bileşenler arasında da yalıtım sağlanmalıdır.

2.1.6. En Zayıf Halkayı Tespit Et ve Güçlendir

Bir yazılımın ilk saldırıya uğrayacak noktası, yazılımın en zayıf noktasıdır. Yazılımın güvenliği en zayıf halkanın güvenliği kadardır. Güvenlik analizi adımımda en zayıf halka tespit edilmelidir. Bu halka kendisine yönelik riski karşılayabilecek şekilde yeniden tasarlanıp gerçekleştirilmelidir.

2.1.7. Saldırı Yüzey Alanını Azalt

Saldırı yüzey alanı, bir yazılım için yetkisiz bir kullanıcının (saldırgan) yazılım ortamından veri alabileceği, veri girebileceği veya yetkisiz işlem yapabileceği noktaların tümüdür. Uygulamaya gereksiz özellikler eklenmemelidir. İletişim yapılacak noktalar sınırlanmalıdır. Saldırı yüzeyi tasarım aşamasında belirlenmeli ve sürekli olarak izlenmelidir.

2.1.8. Savunma Derinliği Oluştur

Savunma derinliği, savunma mekanizmalarının peş peşe uygulanmasından oluşur. Bu şekilde bir savunma mekanizması aşıldığında, sistem tamamen savunmasız kalmaz. Bu bağlamda aşağıdaki hususlar dikkate alınmalıdır:

- Farklı üreticilerin/geliőtiricilerin işletim sistemi, yazılım kütüphaneleri, güvenlik yazılımlarının kullanılması
- Yazılımdaki güvenlik mekanizmalarının, sistemdeki güvenlik mekanizmalarıyla entegrasyonu
- Güvenlik politikalarının tüm bileşenlerde dikkate alınması ve zorunlu olarak uygulanmasının sağlanması

2.1.9. Basit Güvenlik Mekanizması Tasarla

Güvenlik mekanizmaları mümkün olduğunca basit olmalıdır. Bu şekilde hata olasılığının daha az olması sağlanır, hatalar oluştuğunda anlaşılması ve düzeltilmesi kolaylaşır. Bu kapsamda özellikle arayüzlerin mümkün olduğunca basit ve dokümante edilmiş olması gereklidir.

2.1.10. Anlaşılabilir ve Kolay Kullanılabilir Güvenlik Mekanizması Tasarla

Yazılımda kullanıcıyla etkileşim gerektiren güvenlik mekanizmaları anlaşılabilir ve kolay kullanılabilir olmalıdır. Aksi takdirde kullanıcı güvenlik mekanizmalarının etrafından dolaşmak için yollar arayabilir. Bu bağlamda aşağıdaki hususlar dikkate alınmalıdır:

- Kullanıcı işini en kolay şekilde ancak en az yetkiyle yapabilmelidir.
- Kullanıcı yapacağı işe ilişkin güvenlik politikalarını tanımlayabilmelidir.
- Kullanıcının izin verdiğini gösteren eylemleriyle yetkilendirme otomatik olarak verilebilmelidir.
- Kullanıcı daha önce verdiği yetkileri geri alabilmelidir.
- Kullanıcı verdiği yetkileri görebilmelidir.

2.2. YAZILIM GÜVENLİĐİNİ SAĐLAMA YÖNTEMLERİ

Yazılımdaki hataları azaltmak ve kalitesiyle güvenliĐini artırmak için birçok teknik yöntem mevcuttur. Bu bölümde beŐ ana baŐlık altında, güvenlik odaklı olarak kullanılabilcek temel yöntemler açıklanmıŐtır.

2.2.1. Sezgisel (Heuristic) ve Benzetime (Simulation) Dayalı Yöntemler

Bu tür yöntemlerin arasında yer alan statik analiz ve sembolik koŐturma yöntemlerinde, yazılımın alıŐtırılan halini yansıtan bir modeli veya benzetimi oluŐturulur ve bilinen yazılım hata türleri bu model üzerinde aranır. Dinamik analiz, negatif ve rastgele test yöntemlerinde ise doĐrudan yazılımın koŐturulması sırasında gerek ortamın benzetimi oluŐturularak yazılımın bazı hatalara zorlanması saĐlanır. OluŐan hatalar analiz edilir ve genelleŐtirilmiŐ sonuçlara ulaŐılmaya alıŐılır.

Statik Analiz (Static Analysis)

Statik analiz, bir yazılımın belli baŐlı özelliklerini onu alıŐtırmadan analiz etmeyi amalayan bir yöntemdir. Sezgisel (heuristic) yöntemler, güvenilir (sound) analizlere göre daha hızlıdır ancak mantıksal tümevarım yöntemlerinin sahip olduĐu güvenceyi saĐlayamazlar. Yazılımın birçok gösterimi statik analize tabi tutulabilir (gereksinim, mimari, kaynak kodu, alıŐtırılabilir dosya gibi) ancak bunların arasından en olgun yöntemler kaynak kodu statik analiz yöntemleridir.

Statik analiz yazılımlarını karşılaŐtırırken hangi tür yazılım hatalarını özebildikleri harici kaynaklardan doĐrulanmalıdır. ÖrneĐin Carnegie Mellon Üniversitesi'nin <https://www.securecoding.cert.org> sitesinde C/C++, Java gibi dillere iliŐkin yazılım kodlama hataları ve bu hataların hangi araçlarla ne oranda bulunabildiĐine iliŐkin detaylı incelemeler sunulmaktadır.

Sembolik KoŐturma (Symbolic Execution)

Statik analizle yazılımdaki bazı hataların bulunması mümkün olmaktadır. Ancak statik analizlerle kodun derinlerde gizli hataların bulunması çoĐu zaman mümkün olmamaktadır. Bunun nedenlerinden biri statik analiz ürünlerinin yanlış alarm oranını azaltmak ve analizi basitleŐtirerek bazı tür hataları analiz kapsamı dıŐında bırakmasıdır.

Sembolik koŐturma yönteminde yazılımın kodu üzerinde bir soyutlama yapılarak tüm olası alıŐtırmalar modellenmektedir. Yazılım testi kullanılarak sadece tanımlanan akıŐ kapsamındaki hatalar bulunabilir. Sembolik koŐturma yöntemi, test yöntemini

genelleőtirmekte ve daha fazla akıő analiz etmektedir. Sembolik koőturmada bir sembolik koőturma ağacı oluőturulur. Bu ağaç üzerindeki her bir yol birçok gerçek program çalıőma akıőına eődeğerdır. Her bir yol sanki program koőturuluyormuő gibi iőttilir. Bu sayede programı koőturmadan yazılım testi yöntemine göre daha fazla kapsama saėlanabilmektedir.

Dinamik Analiz (Dynamic Analysis)

Dinamik analiz, uygulamadaki açıklıkları tespit etmek için çalıőan yazılım üzerinde ve çalıőma ortamında gerçekleőtirilen analiz araç ve yöntemleridir. Dinamik analiz araçları ve yöntemleri Őu Őekilde özetlenebilir:

- **Aė tarayıcısı:** Aė üzerindeki cihazlar ile yazılımları belirleyen ve bunların özelliklerini tespit eden araçlardır. Örnek: nmap
- **Aė dinleyicisi:** Aė üzerindeki trafiėi dinleyerek anlık analiz eden veya sonradan analiz edilmek üzere kaydeden araçlardır. Örnek: wireshark
- **Aė zayıflık tarayıcısı:** Bir aė cihazına, yazılımına veya aė üzerindeki bir servise daha önceden oluőturulmuő aė trafiėi göndererek güvenlik politikalarına uygunluėunu veya bilinen zayıflıkları tespit etmekte kullanılan araçlardır. Örnek: OpenVAS
- **Bilgisayar tabanlı zayıflık tarayıcısı:** Kurulum ve iőtletim sırasındaki yazılım ile ilgili güvenlik açıklıklarını tespit etmek için bilgisayar üzerindeki yapılandırmaları ve denetim araçlarını test eden araçlardır. Örnek: OpenSCAP
- **Uygulama zayıflık tarayıcısı:** Belirli bir tür uygulamaya girdi üreterek bilinen veya bilinmeyen zayıflıkları tespit etmeye yarayan araçlardır. Örnek: Web uygulaması/servisi zayıflık tarayıcısı (OWASP ZAP, Nikto, Vega vb.), Veritabanı zayıflık tarayıcısı (sqlmap, Scuba)

Negatif Test (Negatif Testing)

Hatalı veya güvenlik kurallarına uymayan durumların ve istisnaların yazılım tarafından kontrol edildiėinin test edilmesine negatif test yöntemi denir. Örneėin; en az 8 karakterlik bir parola gerektiren bir sayfanın 7 karakterli parola ile test edilmesi ve sonuçta yazılımın bu hususta hata mesajı vermesi

Rastgele Test (Fuzz Testing)

Bir yazılıma geçersiz, beklenmeyen veya rastgele veri girdisinde bulunarak istenmeyen davranışta bulunup bulunmayacağını tespit edilmesine rastgele test yöntemi denir. Birçok uygulama zayıflık tarayıcısı aynı zamanda rastgele test kabiliyetlerine de sahiptir.

2.2.2. Biçimsel (Formal) Yöntemler

Biçimsel (formal) yöntemler matematik ve mantık temelli yazılım analiz yöntemlerini ifade eder. Bu kapsamda teorem doğrulama, model doğrulama, ayrıştırma (parsing), tip denetlemesi (type checking), doğruluk ispatları, model tabanlı geliştirme ve doğrulanmış kod tabanı gibi yöntemler kullanılabilir. Biçimsel yöntemleri kullanarak belirli tür yazılım hatalarını tamamen engelleyebilmekte, sürekli test ve hata düzeltmeden oluşan maliyetler azaltılabilmektedir.

Biçimsel yöntemler özellikle otomotiv, demiryolu, havacılık gibi sektörlerde yazılım geliőtirmek için etkin olarak kullanılmaktadır. Örneğin DO-178C havacılık yazılım kalite standardının DO-333 ekinde biçimsel yöntemlerin yazılım doğrulamasında kullanımı ele alınmaktadır. Ayrıca kriptografik protokollerin doğruluk analizlerinde otomatik ispat yöntemleri yaygın olarak kullanılmaktadır.

Biçimsel Program Analizi

Statik analizlerin matematik ve mantık temelli olarak gerçekleştirilmesine biçimsel program analizi denir. Statik analiz yöntemlerinin biçimsel betimleme ve analiz yöntemleriyle beraber kullanımı son zamanlarda önem kazanmıştır. Nükleer enerji tesisleri, ulaőtırma, havacılık, savunma gibi sektörlerde statik analiz ve biçimsel yöntemlerin kullanılmasıyla geleneksel yazılım geliştirme yöntemlerine göre ciddi anlamda yazılım hatası sayısında düşüşler elde edilebilmektedir.

Model Doğrulayıcılar

Yazılımda mevcudiyeti tespit edilmesi istenen gizlilik, kimlik doğrulama, erişim denetimi, bütünlük gibi güvenlik özelliklerinin bir yazılımda mevcut olup olmadığının doğrulanması için kullanılan yöntemdir.

Mantık Yöntemleri

Kodun içerisinde yapılan varsayımların ve beklenen sonuçların mantıksal ifadelerle tanımlanmasından sonra otomatik doğrulayıcı araçlar yardımıyla beklenen sonuçların doğrulanması yöntemidir. Kodun içerisine yerleştirilmiş mantıksal ifadelerden test durumları

üretilebilmektedir. Bu Őekilde hatalar çok daha erken tespit edilebilmekte ve hatanın yeri büyük dođrulukla belirlenebilmektedir.

Model Tabanlı Geliőtirme

Yazılımın güvenlik davranıŐı uygulama alanına özel üst seviye bir dille (Domain Specific Language-DSL) tanımlanarak kaynak kodun büyük bir kısmının veya tamamının oluŐturulan modelden üretilmesi yöntemidir. Önceden güvenliđi dođrulanmıŐ kodlar kullanıldıđında, yazılımcının yeni bir güvenlik açığına neden olma olasılıđı düşer. Ayrıca test durumlarının da tanımlanan modelden üretilmesi mümkün olabilmektedir. Daha çok gömülü sistemlerde uygulanan bir yöntemdir. Kullanıcı arayüzü yoğun yazılımlar için bu yöntemin kullanımı uygun görülmemektedir.

DođrulanmıŐ Araç ve Kod Depoları Kullanma

Yazılımcılar genellikle ihtiyaç duydukları genel amaçlı işlemler için (sıralama, veritabanı erişimi, web protokolleri iletişimi, kriptografik işlemler vb.) İnternet'te paylaşılan kod parçacıklarını kullanmaktadır. Çođunlukla da bu kod parçacıkları herhangi bir güvenlik testine veya gözden geçirmeye tabi tutulmadan yazılımın kod deposuna eklenmektedir. DođrulanmıŐ araçları, güvenlik testi ve gözden geçirmesine tabi tutulmuş yazılım kütüphanelerini, yeniden kullanılabilir gereksinim ve tasarım dokümanlarını içeren ortak bir kütüphanenin oluŐturulması yazılım geliştirme sürecinin güvenliđine olumlu katkı sağlamaktadır. DođrulanmıŐ kütüphanelerin oluŐturulması çabalarına bir örnek olarak, OWASP ESAPI projesi verilebilir (OWASP ESAPI). OWASP ESAPI projesi kapsamında OWASP Top 10 zayıflıkları OWASP Uygulama Güvenliđi Dođrulama Standardına uyum sağlamak için gerekli dil bađımsız arayüzleri ve belirli dillerde referans kütüphaneler geliőtirilmiŐtir.

Kod Güçlendirme

Önceden geliőtirilmiş bir yazılımın güvenlik bakıŐ açısı ile kaynak kodlarının, yapılandırma dosyalarının, derleyicisinin, dinamik kütüphanelerin ve özel amaçlı diđer kütüphanelerin biçimsel yöntemlerle dođrulanması ve güçlendirilmesidir.

İspat TaŐıyan Kod (Proof -carrying Code)

Uygulamanın çalıştırılabilir kodunu takip ederek biçimsel yöntemlerle analiz ederek kodun güvenli Őekilde çalışıp çalışmayacağına karar veren yöntemdir. Yöntem kritik güvenlik ihtiyaçları olan yazılımlarda uygulanmaktadır. Daha maliyetli ve işgücü açısından ihtiyacı yüksek yöntemlerdir. Ancak kritik güvenlik ihtiyaçları olan yazılımlarda geliştirme zamanında

maliyet ve konuşlandırma sonrası deęişiklik ihtiyacının azaltılması nedeniyle maliyet etkinlięi de sağlayabilmektedir.

2.2.3. Yazılım Çalışma Ortamı Güvenliğini Sağlayan Yöntemler

Bazı güvenlik işlevlerinin sistem tarafında ortak kullanılabilir mekanizmalarla sağlanması, saldırılara karşı dayanıklı ve güvenlik hizmetlerini varsayılan olarak sağlayan çalışma ortamı oluşturulması, yazılımın güvenliğini artıracak önlemler arasındadır. Ortak güvenli mimari örüntülerinin kullanılması yeni nesil yazılım teknolojilerinde güvenlięi artırmaktadır. Yazılım çalışma ortamının güvenliğini sağlayacak ortak güvenlik mimari örüntüleri arasında, uygulama konteyneri (container) kullanımı, servis tabanlı mimaride mikro-servisler ile güvenlik işlevlerinin gerçekleştirilmesi ve özellikle bulut ortamında yaygın olarak kullanılan sanallaştırma ortamlarının güvenliğini sağlanması yer almaktadır.

Uygulama Konteyneri (Application Container) Kullanımı

Konteyner, üzerinde çalıştığı bilgisayarın bazı kaynaklarını, kendi içerisinde çalışan uygulama veya sistem için yalıtılan bir nesnedir. Bir konteyner, bağımsız bir bilgisayar veya bütün bir sanal makinenin yalıtım özelliklerini sağlayabilmesine rağmen kısa sürede başlatılabilir özelliğine sahiptir. Genellikle bir sanal makinenin ihtiyacı olan kaynaktan çok daha az kaynakla çalışabilmektedir. Konteynerler tek bir uygulamayı paket olarak yalıtılabildiğinden, bir programı en düşük yetki prensibine göre en düşük kaynak erişimiyle çalıştırabilmek mümkün olmaktadır.

Konteynerler, yeterince yalıtıldığında yazılım zayıflıklarını ciddi şekilde azaltabilmektedir. Bu nedenle konteynerlerin kritik altyapı bileşenlerinin (örneğin Linux çekirdeğindeki denetim grupları ve ad alanları) zararlı girdilere karşı dayanıklı olduğundan emin olunması gerekmektedir. Konteyner yapılandırmalarının modellenmesi, analiz edilmesi, sınanması ve doğrulanması kritik öneme sahiptir.

Mikro Servis Mimarisi İle Yazılımların Geliştirilmesi

Yazılımın küçük servisler halinde ve her bir servisin kendi süreci içerisinde birbirleri ile konteynerler ile iletişim kuracak şekilde tasarlanmasına mikro servis mimarisi denir. Bu mimari ile tasarlanan bir yazılımda yer alan her bir servis de mikro servis olarak adlandırılır. Her bir mikro servis bir iş veya süreç kabiliyetini gerçekleştirir. Bir mikro servis diğer servislerden bağımsız olarak yenileri ile değiştirilebilir, kolaylıkla güncellenebilir ve farklı programlama dilleri ile geliştirilebilir.

Güvenliđi sađlamak veya kontrol etmek için tanımlanan mikro-servislerin her zaman çalışacak, müdahalelere karşı dayanıklı olacak ve doğrulanmış yazılım (reference monitor) olarak geliştirilmesi ile güvenlik açısından ilgili işlevleri kendini yöneten ve doğrulanmış birimler haline getirmek mikro-servislere dayalı güvenli yazılım geliştirme sürecinin önemli bir aşamasıdır.

Sanallaştırma Ortamlarının Kullanılması

Sanallaştırma, tek bir donanım ya da fiziksel sunucu üzerinde birden çok program yığınının (computing stack) sanallaştırmaya özğü bazı yazılımlar ve donanımlar sayesinde çalıştırılmasını sađlayan bir çözümdür. Bu şekilde çalıştırılabilen program yığınları bazen uygulamalar, bazen işletim sistemleri, bazen ise birbirinden farklı görevler üstlenmiş sunucular olabilir. Sanallaştırma, ortak erişilen kaynaklara farklı kanallardan erişme prensibini hayata geçirmek için kullanılacak yöntemlerden biridir.

Sunucuları sanallaştırma teknolojisi, oldukça kolay ve hızlı uygulama imkanları sunmaktadır. Fakat klasik yöntemlerle uygulanan sunucularda karşılaşılan güvenlik risklerinin yanı sıra yeni güvenlik risklerini de beraberinde getirmektedir. Bu nedenle sanallaştırma çözümlerinde mimari, hipervizör ve yapılandırma kapsamında güvenlik gözden geçirmesi ve test yöntemleri uygulanmalıdır.

2.2.4. Birden Fazla Araçla Analiz Yöntemleri

Yazılım güvenlik ve kalite araçları, yazılımlarla ilgili çok çeşitli veri üretmekte ve depolamaktadır. Mevcutta birçok ticari ve açık kaynaklı araç bulunmaktadır. Ancak bu araçlarda veri standardı olmaması nedeniyle farklı analiz araçları arasında bilgi aktarımında sorunlar oluşabilmektedir. Örneğin bir yazılım tarafından yazılım hatası olarak belirtilen bir hata, bir diđerinde zayıflık olarak nitelendiğinde, bu hatanın derecesinin yorumlanmasına ilişkin tutarsızlıklar ortaya çıkabilmektedir. Birden fazla araç ihtiyaçlar doğrultusunda aşağıdaki yöntemlerle kullanılabilir.

- Tüm araçların çalıştırılarak araçların çıktılarının ortak bir formata dönüştürülerek incelenmesi
- Tüm araçların çalıştırılarak her bir araçtan aynı zayıflık için birbirini destekleyen veya gelişen sonuçların incelenmesi
- Farklı zayıflıkları tespit etmek için hangi araçların kullanılacağına belirlenmesi ve ilgili zayıflığa göre ilgili aracın çalıştırılarak sonuçların incelenmesi

2.2.5. Yazılım Güvenliđini Sađlamada Diđer Yöntemler

Programlama Dilleri ile Güvenliđinin Artırılması

Yazılım geliőtirirken sadece yazılımdan beklenen işlevselliđi sađlamak yeterli deđildir. Yazılımı güvenli hale getirmek için yazılım iđerisindeki hataları çözmek tek başına amaca ulaşmasını sađlayamaz. Yazılımın kendinden beklenenleri tam anlamıyla yapmasının dıőında yapması istenmeyen Őeyleri de ihtimali ne kadar küçük olursa olsun gerçekteşmesini engelleyebilmesi gerekmektedir. Programlama dili sečilirken dilin güvenlik konusundaki zafiyetleri (veri tipi denetimi, hata yönetim mekanizması vb.) ve güvenliđi sađlamak için sađladığı kabiliyetlerin deđerlendirilmesi gerekmektedir.

C, C++, Java, Perl gibi diller için belirlenmiş kodlama standartları incelenerek projede kullanılacak programlama diline uygun güvenli kodlama standardı oluşturulmalıdır. Güvenlik açıklarına sebep olabilecek kod parçacığı örnekleri yazılımcılarla paylaşarak her bir dil için belirlenen kodlama güvenlik standartlarına uyulmalıdır.

Deđişen Hedef Savunmasının Uygulanması

Bir yazılımın normal çalışmasını bozmadan, yapısının ve özelliklerinin sürekli olarak deđerştirilerek saldırganın zayıflıkları sömürme olasılıđı en aza indirilmeye çalışılmasına deđerşen hedef savunması denir. Buna örnek olarak bellek taşıma saldırılarını engellemek için kullanılan ve yığın belleđini rastgele hale getiren yöntemler (ASLR gibi) gösterilebilir. Her bir bellek ayırma isteđi ayrı ve rastgele bir konuma verilir. Bu Őekilde belleđin alanının dıőına yazma ve okuma yoluyla kritik bilgilerin çalınması veya zararlı yazılım koőturulması ihtimali azaltılmış olur.

Ađ ve Sistem Seviyesinde Önlem Alınması

Her ne kadar yazılım geliőtirme sürecinin parçası olmasa da, yazılımın koőtuđu ortamdaki ađ ve sistem seviyesinde önlemlerin alınması, yazılımdaki açıklıkların sömürülmesini zorlaőtıran bir unsur olduđundan göz ardı edilmemelidir. Uygun gözden geçirme, sızma testi, sıkılaőtırma yöntemlerinin uygulanması gerekmektedir. Bu kapsamda ayrıca ađ ve sistem cihazlarında güvenliđi önceden bilinen yapılandırmalar ve yazılım sürümleri kullanılmalıdır.

İşletim Sistemi Arayüzü Seviyesinde Önlem Alınması

Yazılımın işletim sistemlerine yaptıkları çağrılarının güvenli hale getirilmesi gerekmektedir. İşletim sistemleri ile iletişimde saldırı yüzeyini azaltmak ve zafiyetlerin sömürülme olasılıđını

düşürmek için arayüz seviyesinde savunma derinliđi ve çeşitlendirme uygulanması gerekebilmektedir.

2.3. GÜVENLİ YAZILIM GELİŐTİRME YAŐAM DÖNGÜSÜ

Güvenli bir yazılımın tasarımı, gerçeklenmesi, yapılandırılması, kurulması ve desteklenmesi, yazılımın birçok saldırıya karşı doğru bir şekilde çalışmaya devam edecek, karşı koyamadığı saldırıların etkisini sınırlamak için gerekli önlemleri içinde barındıracak ve kısa sürede normal çalışmaya devam edecek şekilde olabilmelerini sağlayacak bir süreçte hayata geçirilir. Bu tür bir süreç güvenli yazılım geliştirme yaşam döngüsü olarak adlandırılır. Güvenli yazılım geliştirme yaşam döngüsünde;

- Zafiyetler ve zayıflıklar geliştiriciler tarafından ilk elden engellenir. Bu amaçla süreçte geliştiricilerin güvenli yazılım eğitimini almış olmaları kritik öneme sahiptir.
- Yazılımın saldırılara karşı dayanıklı, saldırılar altında çalışmaya devam edebilecek toleransa ve sürekliliđe sahip olması sağlanır. Bu amaçla yazılım güvenlik gereksinimleri en baştan tespit edilerek, tehdit analizi yapılır, güvenli tasarım prensipleri uygulanır ve sonrasında yapılacak güvenlik testleriyle gereksinimlerin sağlanıp sağlanmadığı belirlenir.
- Kötü niyetli geliştiricilerin bilerek ve isteyerek kodun içerisine zayıflık veya kötü niyetli işlevler dahil etmesi durumu kaynak kod analizi, statik analiz, güvenli kurulum gibi önlemlerle engellenebilir.
- Yazılım ve çevresindeki ortam arasındaki etkileşimlerden dolayı ortaya çıkabilecek zafiyetleri en aza indirmek için, geçmiş zafiyetlerin incelenmesi, dış inceleme, sızma (penetrasyon) testi ve olay müdahalesi gibi önlemler uygulanır.

1.1.1 Geliştiricilerin Güvenli Yazılım Geliştirme Konusunda Eğitilmesi

Yazılım güvenliđi genelde kar sağlamayan bir kısmı olarak görölmektedir. Bu nedenle geliştiricilerin güvenli yazılım geliştirme konusunda eğitim alması öncelikler arasında yer almayabilmektedir. Güvenlik teknolojilerine yatırım yapılması daha öncelikli göröldüğünden mevcut kaynak bu alana yönlendirilir. Ancak yazılım uygulamalarını bir saldırı sonrasında düzeltmek, finansal ve itibar kayıplar açısından çok yüksek maliyetli olmaktadır. Geliştiricilerin eğitimi ise, saldırılara karşı proaktif olarak önlem alınmasını sağladığından güvenliđin toplam maliyetini düşüren bir yaklaşımdır.

Geliştiriciler özellikle de yeni geliştiriciler; tehdit modelleme, güvenli yazılım tasarım prensipleri, bu prensiplerin nasıl uygulanacağını gösteren kod örnekleri, programlama

dillerindeki güvenlik açıklıkları ve bu açıklıkların nasıl kapatılacağını gösteren örnekler ile yazılım güvenlik testleri konusunda eğitime tabi tutulmalıdır. Ayrıca bu eğitimlerin uygulamalı olması ve eğitim sonunda bir deneme projesiyle bilginin pekiştirilmesi sağlanmalıdır.

1.1.2 Yazılım Güvenlik Gereksinimlerinin Tanımlanması

Güvenlik gereksinimleri, güvenlik risklerinin tanımlanması için ne yapılacağını veya hangi özelliklerin sağlanacağını belirten gereksinimlerdir. Güvenlik gereksinimleri risk analizinin önemli bir parçasıdır ve birçok standartta yapılması önerilen bir faaliyettir.

Güvenlik gereksinimlerinin tanımlanması iki ana aşamada gerçekleştirilir (Şekil 3). İlk aşamada risk analizi çalışmasıyla Güvenlik Risk Analizi Dokümanı üretilir. Daha sonra risk analizi çalışmasının çıktıları kullanılarak güvenlik gereksinimleri türetilir. Güvenlik gereksinimleri ile Güvenlik Gereksinimleri Belirtim Dokümanı veya Sisteme Özgü Güvenlik Gereksinimleri Belirtim Dokümanı oluşturulur.



Güvenlik Risk Analizi

Güvenlik risk analizi güvenlik gereksinimlerinin belirlenmesi öncesi önemli bir faaliyettir. Bu faaliyet, varlık analizi ve tehdit/zayıflık analizi olmak üzere iki adımdan meydana gelir.

Varlık analizinde ilk olarak varlık envanteri belirlenir. Yazılım geliştirme sürecinde bulunan varlık türlerine aşağıdakiler örnek olarak verilebilir:

- Bilgi varlıkları
- İş kuralları

- Servisler veya metotlar
- Yazılım kodu
- Firmaya özgü formüller
- Şifreleme yöntem ve anahtarları
- Veritabanları
- Kişiler veya kişilerin sahip olduđu belirli bilgi ve yetenekler
- Hesap bilgileri ve hesapla ilişkilendirilmiş fonlar
- İş kayıtları

Varlıkların tespit edilmesinde Veri Akış Şemalarının (Data-Flow Diagram) oluşturulması yönetimi uygulanabilir. Bu yöntemde süreçler, harici öğeler, veri depoları, veri akışları ve güven sınırları belirlenir. Süreçler, harici öğeler ve veri depoları arasındaki akışlar hiyerarşik olarak DFD diyagramları yoluyla tanımlanır.

Varlıklar tespit edildikten sonra Varlık Değerlendirme (Asset Valuation) adımı gerçekleştirilir. Bu adımda matris, Delphi yöntemi, Lineer değerlendirme (ISO 27005) gibi yöntemler izlenebilir. Şekil 4'te bir örnek yazılım varlıkları değerlendirme matrisi gösterilmiştir. Bu matriste, bir önceki adımda belirlenen varlıklar, maddi değer ve iş sürecine etki olmak üzere iki boyutta sınıflandırılmıştır. Her iki boyut için de, Düşük-Orta-Yüksek olmak üzere üç seviye belirlenmiştir. Yazılımın türü ve varlık envanterindeki varlıkların değer farklılıklarına göre, eksenlerdeki boyutlar ve boyutlardaki seviye sayısı farklı olarak belirlenebilir.

Maddi değer?	Düşük	Orta	Yüksek
İş sürecine etkisi?			
Düşük	(Genel) Bilgi varlıkları	İş kayıtları	Hesap bilgileri
Orta	İş Kuralları	Yazılım Kodu	Kişiler Müşteri veritabanları
Yüksek	Kullanıcı bilgileri veritabanı	Yazılım Servisleri Şifreleme yöntem ve anahtarları	Firmaya özgü formüller

Şekil 4 Yazılım Varlıkları Değerlendirme Matrisi

Yazılım varlıkları belirlendikten sonra bunların üzerindeki saldırıların etkisi değerlendirilmelidir (Impact Assessment). Bir tehdit bir veya daha fazla varlığı etkileyebilir veya bir varlığı kısmen etkileyebilir. Dolayısıyla etki değeri saldırıdan etkilenecek varlıkların değerlendirmesini girdi alan bir fonksiyon ile hesaplanır. Bu amaçla etkilenecek varlıkların içindeki en değerli olan (MAX) veya tüm etkilenen varlıkların toplamı (SUM) fonksiyonu kullanılabilir.

Bir sonraki adımda tehdit ajanları belirlenmeli ve tehdit sınıflandırma (tehdit profili belirleme) yapılmalıdır. Tehditler insan kaynaklı veya insan dışı olabilir. İnsan kaynaklı tehditler, kasıtsız, kasıtlı ve teknik tehditlerdir. İnsan dışı tehditler ise, doğal ve çevresel tehditlerdir. Tehditleri sınıflandırmak için ENISA tarafından tanımlanan yöntem veya STRIDE yöntemi kullanılabilir. Örnek olarak Şekil 5'te ENISA sınıflandırmasına uygun bir tehdit profili değerlendirme matrisi gösterilmektedir.

Kaynak ----- Yetenek/Teknik Zorluk	Düşük	Orta	Yüksek
Düşük	Meraklı çocuk	Siber savaşı	Siber terörist
Orta	Çevrimiçi sosyal hacker	Siber suçlu	Kötü niyetli geliştirici/operatör
Yüksek	İç tehdit	Hacker organizasyonları	Siber casusluk

Şekil 5 Tehdit Profili Değerlendirme Matrisi

Tehdit profili değerlendirmesi sonucunda kategoriler belirlenerek Tehdit ajanları çıkarılır:

1. Kategori A (Turuncu) -> Tehdit derecesi = Düşük
2. Kategori B (Mavi) -> Tehdit derecesi = Orta
3. Kategori C (Yeşil) -> Tehdit derecesi = Yüksek

Buradan olasılık değeri hesaplanabilir. Son olarak, etki ve olasılık değerleri riski hesaplamak için kullanılabilir. Bu amaçla aşağıdaki formüller örnek olarak kullanılabilir.

- Olasılık = Zayıflık * Tehdit derecesi,
- Etki = MAX (Varlığın kuruma olan değeri, Saldırının dolaylı etkileri)
- Risk = Olasılık * Etki

Bu kısımda genel çerçevesiyle anlatılan yazılım risk analizi için uluslararası kuruluşlar tarafından tanımlanmış belli başlı risk analizi metodolojileri kullanılabilir. Bunlara örnek olarak NIST SP 800-160, ISO 27005, ISO 31000, OWASP Risk Rating Methodology verilebilir.

Güvenlik Gereksinimleri Belirleme

Gereksinimlerin belirlenmesinde ISO 27002'nin tavsiye ettiđi pratikler aŐađıdaki gibi olup bu pratikler güvenlik gereksinimlerinin belirlenmesi için de geçerlidir:

- Gereksinimlerin tanımlanması için politikanın, rol ve sorumlulukların belirlenmesi
- Gereksinimleri belirlemek için yöntemlerin oluşturulması
- PaydaŐlar tarafından gereksinimlerin gözden geçirilmesi
- İş varlıklarına olan etkisine göre gereksinimlerin önceliklendirilmesi
- Gereksinimler için deđişiklik yönetiminin uygulanması
- Ürün kabul kriterlerinin belirlenmesi

Güvenlik gereksinimlerinin tanımlanmasında önemli diđer bir nokta da izlenebilirliđin oluşturulmasıdır. Sistem gereksinimlerinin güvenlik gereksinimlerine, güvenlik gereksinimlerinin tasarıma ve test öğelerine izlenebilirliđi sağlanmalıdır.

EK-2'de tanımlanmış olan Uygulama Güvenlik Kuralları (UGK), güvenlik gereksinimlerinin tanımlanmasında şablon olarak kullanılabilir. Bu amaçla uygulamanın çalışacağı bağlam göz önüne alınıp aŐađıdaki sorulara yanıt verecek şekilde güvenlik gereksinimleri oluşturulabilir.

- Bir eylemi kim gerçekleŐtirecek (kim);
- Riski azaltmak veya ortadan kaldırmak için alınacak önlem (nasıl);
- Önlemin hangi zamanda hayata geçirileceđi (ne zaman);
- Önlemin nerede hayata geçirileceđi (nerede);
- Bu gereksinimin etkilediđi varlık ve bilgi öğeleri (ne);
- Güvenliđin adreslediđi risk ve/veya tehdit ajanı (neden).

Güvenlik Gereksinimleri aŐađıdaki şekilde sınıflandırılabilir:

- **İş güvenlik gereksinimleri:** Kuruluşun iş çerçevesinden en az bir güvenlik risklerinin önlenmesi için tanımlanan gereksinimlerdir. Örneđin müşterinin bilgisini korumak, kuruluşun hedeflerine yönelik risklerin önlenmesi vb.
- **İş güvenlik kuralları:** Kuruluşun iş kurallarına yönelik risklerin en az birisini adresleyen gereksinimlerdir. Örneđin yönergeler, iç kurallar, iş yapıŐ biçimleri vb.
- **Yasal güvenlik gereksinimleri:** Kuruluşun bađlı olduđu mevzuattan kaynaklanan güvenlik gereksinimleridir. Örneđin kanun, yönetmelik vb.

- **Kullanıcı güvenlik gereksinimleri:** Yazılımı kullanıcılarının eylemlerinden kaynaklanan güvenlik risklerini önlenmesi için tanımlanan gereksinimlerdir.
- **Geliőtirme ve iŐletim güvenliđi gereksinimleri:** Yazılım geliőtirme veya iŐletim ortamındaki güvenlik risklerinin önlenmesi için tanımlanan gereksinimlerdir.
- **Kalite öznitelikleri:** Geliőtirilen yazılımın sađlaması gereken kalite özelliklerine iliŐkin güvenlik gereksinimleridir. Örneđin kullanılabilirlik, bütünlük, taşınabilirlik, bakım gereksinimleri vb.
- **Sistem güvenlik gereksinimleri:** Yazılım tarafından sađlanan iŐlev ve özelliklere yönelik risklerin önlenmesi için tanımlanan gereksinimleridir.
- **Süreç güvenliđi gereksinimleri:** IŐletim süreçleri, yazılım kurulumu, bakımı, süreklilik ve arŐivleme gibi yazılım yaŐam döngüsüne yönelik risklerin önlenmesi için tanımlanan gereksinimleridir.
- **IŐlevsel güvenlik gereksinimleri:** Yazılım tarafından gerçekleştirilen kullanım durumlarına yönelik risklerin önlenmesi için tanımlanan gereksinimleridir.
- **DıŐ arayüz güvenlik gereksinimleri:** Web arayüzleri, harici iletiŐim arayüzleri gibi yazılım tarafından dıŐarıya aŐılan arayüzlerden gelebilecek risklerin önlenmesi için tanımlanan gereksinimleridir.
- **Altyapı güvenliđi gereksinimleri:** Yazılımı destekleyen altyapı ortamından kaynaklanan risklerin önlenmesi için tanımlanan gereksinimleridir.
- **Kısıtlar:** Bir güvenlik gereksinimine iliŐkin kısıtlar uygulamanın yapısından kaynaklanan kısıtlamaları içerir. Örneđin belirli iŐlemlerin iki aktör tarafından yapılması gerektiđi, uygulamaya İnternet üzerinden HTTPS protokolüyle eriŐilmesi gerektiđi vb.

1.1.3 Güvenlik Tasarımı ve Mimarisi

Tasarımı gizli tutarak güvenlik yaklaŐımı (Security through obscurity), yeni nesil araç ve yöntemlerle (tersine mühendislik, kod analizi araçları vb.) artık geçerliliđini yitirmiŐtir. Güvenlik tasarımının karşı tarafça bilindiđi varsayılarak güvenlik analizlerinin yapılması gerekmektedir. Bu bağlamda, uygulamanın tehdit modeli tanımlanmalı, güvenlik tasarım dođrulama yöntemleri (tehdit modelleme ve risk analizi, biŐimsel yöntemler, tasarım gözden geçirme vb.) projelerde standart olarak uygulanmalıdır.

Uygulamanın güvenlik iŐlevleri belgelendirilmiŐ olmalıdır. Güvenlik iŐlevlerinin, kullanım durumu, akıŐ diyagramı, etkinlik diyagramı, sınıf diyagramı, gereksinimler diyagramı, blok Őeması gibi modelleri oluŐturularak proje iŐerisinde tekil bir iletiŐim yönteminin

standartlaştırılması yoluna gidilmelidir. Bu şekilde deęişen gereksinimlere de uyum sağlayacak etkin bir belgelendirme sağlanabilir.

Yazılımın güvenlik tasarım aşamasında aŐaęıda belirtilen hususlarda ek çalıŐma yapılıp yapılmayacağına karar verilmelidir:

- GeliŐtirilen uygulamaların mimarisi güvenli yazılım ilkelerine uygun olmalıdır. Uygulamanın üst seviye mimarisi tanımlanmalı, yazılım bileŐenlerinin iŐlevleri belirli olmalı ve gereksiz iŐlevler yazılımda yer almamalıdır.
- Yazılım tasarımında kütüphanelerin ve dil özelliklerinin kullanımına dikkat edilmelidir. Dil çerçevesi tarafından desteklenmeyen eski fonksiyonlar kod deposunda yer almaması sağlanmalıdır. Potansiyel olarak tehlikeli fonksiyonlar için eŐdeęer güvenli fonksiyonların kullanılması sağlanmalıdır.
- Uygulamanın güvenlik katmanları birbirinden ayrılmalı ve güvenlik iŐlevlerinin normal iŐlemlerin gerçekleştirildięi çalıŐma ortamından farklı bir izole ortam içerisinde gerçekleştirilmesi sağlanmalıdır. Bu amaçla, TPM (Trusted Platform Module), TEE (Trusted Execution Environment), HSM (Hardware Security Module), uygulama konteyneri gibi teknolojiler kullanılabilir. Ayrıca servis tabanlı mimari (SOA, mikro servis) tasarım örüntülerinin kullanılması, güvenlik katmanları ve iŐlevlerinin birbirlerine ve dięer yazılım bileŐenlerine baęımlılıęını azaltacağından mimari tanımlanırken tercih edilen bir yaklaŐım olmalıdır.
- Uygulama veritabanında kiŐisel veri içeren birincil anahtar (kimlik no, e-posta adresi vb.) kullanılmamalıdır. KiŐisel veriler üzerinde iŐlem yapılması ana amaç olmayan durumlarda uygulama kiŐisel verileri maskeleyerek görüntülemeli, aktarmalı veya iŐlemelidir. Uygulamanın geliştirme, test ve eęitim ortamlarında kiŐisel veri olmamalı, anonimleŐtirilmiŐ veriler kullanılmalıdır. KiŐisel veri içeren uygulama veritabanı yedekleri Őifreli olarak saklanmalıdır. Uygulama, kiŐisel veriler için saklama sürelerini takip edebilmeli ve saklama süresi sona eren verilerin silinebilmesini / yok edilebilmesini sağlamalıdır.
- Web uygulamalarının tasarımı aşamasında mümkün olduęunca web geliştirme çerçevesi tarafından sağlanan oturum yönetimi iŐlevleri kullanılmalı, özel oturum yönetim bileŐeni geliştiriliyorsa tüm bilinen oturum yönetimi saldırılarına karşı dayanıklı olduęu doęrulanmalıdır.

1.1.4 Güvenli Kodlama

Yazılımlardaki zayıflıklar yetkisiz kişiler tarafından sömürülerek zarara neden olabilir. Yazılım uygulamalarındaki zayıflıkların azaltılmasının önemi birçok kurum tarafından anlaşılmıő ve bu konuda açık kaynaklar oluşturulmuőtur. Geliőtiricilerin yeni zayıflıkları mevcut geliőtirmekte oldukları yazılım projesine dahil etmemelerini saęlamak için çeőtli kaynaklardan derlenmiő kılavuzlar ve denetim listeleri oluşturulmalıdır. Kodlamadan kaynaklı güvenlik açıklıkları ve önlemlerine iliőkin çeőtli kodlama standartları oluşturulmuőtur. Bu standartlardan uyarlama yapılarak kullanılması, projede sürekli ve otomatik olarak çalıőtırılabilmesi saęlanmalıdır.

1.1.5 Güvenli Kurulum

Savunma derinlięi oluştur prensibi doęrultusunda mümkün olduęunca uygulamaların mimarisi tek parça (monolitik) olarak tasarlanmamalıdır. Tek parça yapıda bir mimariye sahip uygulama, saldırganların tek bir uygulama bileőenine baőarılı bir saldırı düzenlemesi durumunda tüm uygulamanın ele geçirilmesi veya işlevsiz bırakılabilmesi anlamına gelecektir. Günümüzde servis odaklı mimarilerde uygulama geliőtirmek için sanallaőtırma ve konteyner teknolojileri mevcuttur. Bu teknolojiler hem uygulama geliőtirme aőamasında hem de, uygulama çalıőtma ortamlarında güvenli bir şekilde, gerekli sıkılaőtırmalar yapılarak kullanılmalıdır.

Çoęu sistem bileőeni ve teknoloji, varsayılan kurulumları yapıldıęında güvenlik ačasından zafiyet yaratabilecek yapılandırma ayarlarına ve her zaman gerekli olmayan işlemlere ve özelliklere sahip olurlar. Derinlemesine güvenlik prensibi dahilinde, saldırı yüzeyini azaltmak ve saldırganların işini zorlaőtırmak amacıyla, bir uygulamaya ait tüm bileőenlerin ve altyapı unsurlarının sıkılaőtırılması gerekmektedir. Bir sistem bileőeninin sıkılaőtırma işlemi bileőene ait gereksiz yazılımları ve özellikleri kaldırmayı ve o bileőene ait yapılandırmada mümkün olan en güvenli sečeneklerin sečilmesini içerir. Sıkılaőtırma işlemi sırasında uygulamanın işlevlerini yerine getirilmesinin engellenmedięine dikkat edilmelidir.

Sıkılaőtırma kuralları o ürünlerin üreticileri tarafından veya çeőtli kurumlar tarafından internette yayınlanmaktadır. Sıkılaőtırma kuralları genellikle bir kontrol listesi şeklinde verilmektedir. Sıkılaőtırma kuralları için mevcut standartlardan biri SCAP standardıdır. Bu standardın kullanımı için açık kaynak kodlu OpenSCAP yazılımı kullanılabilir. OpenSCAP yazılımı her bir sıkılaőtırma kuralı için bir denetim mekanizmasına sahiptir ve çalıőtırıldıęında bu denetimleri teker teker yaparak hangi maddelerin baőarılı olduęu hangi

maddelerin başarısız olduđu bilgisini kullanıcıya vermektedir. Ayrıca her bir denetim maddesi için düzeltici betikler de örnek olarak verilmektedir.

Sistem genelindeki sıkılaőtırma işlemlerinin, merkezi bir şekilde otomatik olarak yapılması, hem sıkılaőtırma işleminin maliyetini düşürme, hem sıkılaőtırmada karşılaşılabilecek insan hatalarını azaltma hem de sıkılaőtırma işlemini uygulama geliştirme sürecinin bir parçası haline getirebilme açılarından faydalıdır.

Uygulama bileşenleri için sıkılaőtırma profilleri önceden tanımlanıp betikler şeklinde hazırlanarak tanımlanacak filtreler aracılığıyla hangi istemcilere uygulanacakları belirlenerek bahsi geçen yapılandırma yönetimi araçları vasıtasıyla uygulanabilirler.

Sıkılaőtırılmış çalışma ortamı oluşturmak için ise yine Puppet, Chef vb. araçlar kullanılabilceđi gibi önceden sıkılaőtırılmış temel kalıp imajlar kullanılarak Vagrant, Docker gibi sanal makine veya konteyner oluőturma araçları kullanılarak altyapı sıkılaőtırılması da yapılabilir. Puppet / Chef kullanmanın getireceđi dezavantaj olarak her bileşene ajan kurulması gerekmesi gösterebilir fakat Puppet / Chef kullanıldığında çalışma anında yapılandırma deđişikliği yapabilmek de mümkün olacaktır.

Kod kalitesi ölçümü, testler, otomatik konuşlandırma ve yapılandırma işlemlerinin merkezi yönetimi için Jenkins vb. araçlar kullanılabilir. Bu araç aracılığıyla bir iş akışı oluşturulup merkezi olarak güvenli konuşlandırma için gerekli araçların gerektiđi sırayla çalışması, gerekli işlemlerin otomatik olarak yapılması yönetilebilir.

1.1.6 Güvenlik Analizleri ve Testleri

Yazılım geliştirme yaşam döngüsünün deđişik aşamalarında kullanılmak üzere yazılım güvenlik analizleri ve testleri kapsamında aŐađıdaki faaliyetlerden proje özelinde uygulanabilir olanların gerçekleştirilmesi gerekmektedir.

Saldırı Modelleme

Sistem mimarisi zayıf yönleri veya zayıf noktaları bulmak için saldırganın bakış açısı ile incelenir.

Kaynak Kod Analizleri

- Uyarı Bayrakları: Yazılımlar ve platformlar için programlamaya dahil edilen uyarı mekanizmalardır. Kaynak kodunu işlerken tehlikeli durumlara karşı uyarıda bulunurlar.

- Kaynak Kod Kalite Analizi: Kaynak kodu kalite analiz araçları, yazılım kaynak kodunu inceler ve kötü kodlamanın, kötü mimari ve kodlama örüntülerinin tespiti için tarama yapar. Kötü kodlama, bağlama bağı olarak bozuk işlevselliğe, düşük performansa, yüksek bakım maliyetine ve güvenlik zayıflıklarına yol açabilir.
- Kaynak Kod Zafiyet Analizi: Kaynak kodu zayıflık analiz araçları, yazılım kaynak kodunu incelemekte ve bilinen açıklık türlerine karşı örüntü eşleřtirmeleri kullanarak zayıflıkları aramaktadır. Bu tür araçlar için "kaynak kodu güvenlik analiz aracı", "statik uygulama güvenlik testi aracı", "statik analiz kod tarayıcısı" veya "kod zayıflığı analiz aracı" gibi isimlendirmeler de mevcuttur.
- Kaynak Kodundan Mimari, Tasarım ve İşlevlerin Çıkarılması: Bu araçlar, analize yardımcı olmak için kaynak kodundan mimari ve tasarıma ilişkin bilgileri çıkarır. Bu araçlar, kaynak kodu kalite veya zayıflık analizi yapmak için temel olarak kullanılabilir.

İkili Kod / Sekizli Kod Analizleri

- Geleneksel Virüs Tarayıcıları: Geleneksel virüs tarayıcıları ikili kod veya sekizli kod içinde imza taraması yapmaktadır. Bazı modern virüs tarayıcılar davranış analizi de yapmaktadır.
- Kalite Analizi: İkili kod veya sekizli kod kalite analizi araçları kaynak kod kalite analizi ile aynı işlemi ikili koda veya sekizli koda bakarak yapmaktadır.
- Sekizli Kod Zafiyet Analizi: Sekizli kod zafiyet analizi araçları kaynak kod zafiyet analizi ile aynı işlemi sekizli koda bakarak yapmaktadır.
- İkili Kod Zafiyet Analizi: İkili kod zafiyet analizi araçları kaynak kod zafiyet analizi ile aynı işlemi ikili koda bakarak yapmaktadır.
- Uygulamalar Arası Akış Analizi: Bu araçlar haberleşme arayüzlerini ve izinleri tespit ederek, uygulamalar arası veri ve kontrol akışını inceleyip güvenlik politikalarını ihlal eden akışları kontrol eder.
- İkili Kod/Sekizli Kod Basit Bilgi Çıkarma: Kod içindeki karakter katarlarını bulma gibi işlevlere sahip araçlardır.
- İkili Kod/Sekizli Kod Üzerinden Uygulamanın Kullandığını Beyan Ettiği Hakları Kontrol Etme: Kodun sadece beyan ettiği hakları kullandığını test etmede kullanılan araçlardır. Bu haklar uygulamaya verilen hakları kapsar, dosyalar veya hafıza üzerine verilen haklar bu kapsam dışındadır.

Karartılmıő Kod Tespiti

Karartılmıő kod tespit araçları ile karmaőıklaőtırılmıő, okunmaz hale getirilmıő kodların belirlenmesi yapılır. Bu araçlar, kaynak kod ikili kod veya sekizli kod dosyalarına uygulanabilir. Kod karartma tersine mühendisliđi engellemek için kullanılmakla beraber kod üzerinde analiz yapmayı zorlaőtırmaktadır.

İkili kod / Sekizli Kod Çözücülerini (Disassembler) İle Analiz

İkili / Sekizli kodu çözücülerini, sayılardan oluőan makine dilindeki kodları insanlar tarafından anlaşılabilir sözel komutlara dönüőtürür; bu komutlar insanlar veya otomatik araçlar ile analiz edilebilir.

İnsan Tarafından Gözden Geçirme

- Gözden geçirme genellikle kaynak kodu ile yapılır ancak makine kodu ile de yapılabilir (çođunlukla yukarıda belirtildiđi gibi ikili veya sekizli kod çözücüsü kullanılır). İnsan incelemeleri, gereksinimlere, mimariye, tasarıma ve test bulgularına da uygulanabilir.
- Odaklı Manuel Kontrol: Bu yöntemde, belirli soruları cevaplamak için manuel kod analizi yapılır. Genelde 100 satırlık koddan daha azı için uygulanır. Örneđin, yazılım gerektiđinde yetkilendirme gerektiriyor mu? Yazılım arayüzleri girdi denetimi yapıyor mu?
- Manuel Kod Gözden Geçirme: Bu yöntemde kodun manuel olarak incelenmesi yapılır. Örnek olarak kodun içinde zararlı bir kodun bulunup bulunmadıđına bakılabilir
- Denetimler: IEEE 1028 denetimi, yazılım anormalliklerinin tespiti ve tanımlanması için dikkatlice yapılan incelemeleri içerir.
- Üretilmiő Kod Denetimleri: Bu yöntemde, üretilen ikili veya sekizli kodun, kaynak kodunu dođru bir şekilde temsil edip etmediđini belirlemek için inceleme yapılır. Örneđin, bir derleyici veya sonraki bir süreç yazılıma kötü niyetli bir kod eklerse, bu yöntemle bu algılanabilir. Bu denetimler genellikle kodun tümü yerine belli bir bölgesinde gerçekleştirilir.

Güvenli Platform Seçimi

- Güvenli Diller: Kullanılabilir diller içinde daha güvenli olanları seçilerek, güvenlik açığı oluőmasını engellemek veya daha zor hale getirilmesi amaçlanır.
- Güvenli Kütüphane Seçimi: Güvenli kütüphaneler, güvenli uygulama geliştirilmesi kolaylaőtıran mekanizmalar sađlar. Bu tip kütüphaneler tek başına kullanılabilirler gibi daha büyük platformların bir parçası da olabilirler.

- Güvenli İŐletim Sitemi: Güvenli iŐletim sistemi, açıklıkları ve zafiyetleri azaltmak için sıkılaŐtırılan bir iŐletim sistemi ve platformudur.

Kaynak Analizi

Kaynak analizinde, nereden geldiđini belirlemek için kaynak kodu, sekizli kodu veya ikili kod analiz edilir. Bu analizle, kullanımı riskli olan eski ve güvensiz kütüphaneler ile yeniden kullanılan kodlar belirlenir.

Dijital İmza Doğrulama

Dijital imza doğrulama ile yazılımının yetkili kaynaklardan geldiđi ve geliŐtirme sonrasında deđiŐtirilmediđi doğrulanır. Bu iŐlem kriptolojik imzaların kontrol edilmesiyle yapılır.

Yapılandırma Kontrolü

Yapılandırma kontrolü ile, güvenlik gereksinimleri de dahil olmak üzere yazılımın tüm gereksinimlerini karŐıladıđından emin olmak için yazılım yapılandırmasını incelenir. Yapılandırma, yazılımın nasıl eriŐildiđini, nasıl korunduđunu ve nasıl çalıŐacađını belirleyen ayarlar kümesidir.

İzin Bildirimi Analizi

İzin bildirimi analizi araçları yapılandırma kontrolü yapan araçlara benzer şekilde çalıŐır. Android uygulamaları gibi izin bildirimi yapan uygulamaların izin bildirimleri ve olası risk seviyeleri incelenir. Bu analizde koda bakılmamaktadır.

Sürüm Kontrol Araçları

Sürüm kontrol araçları, uygulamada hangi deđiŐikliđi kimin yaptıđını ve ne zaman yapıldıđını kaydeder ve izler. Bu bilgi, kimlerin uygulamada güvenlik açıklıđı yaratacak kodu (kasıtsız veya kötü amaçlı) eklediđini tespit etmeyi kolaylaŐtırabilir. Sürüm kontrolü, açıklık yaratacak kötü kod için bir caydırıcılık ve iyileŐtirme için bir baŐlangıç noktası oluŐturur.

Kod Karartıcı

Kod karatma araçları, kaynak kodu, sekizli kodu veya ikili kodu alır anlaŐılması veya üzerine tersine mühendislik yapılması zor bir hale getirir.

Yeniden Derleme ve KarŐılaŐtırma

Yeniden derleme ve karŐılaŐtırma tekniđi, belirtilen bir kaynak kodundan tekrardan bir sekizli kod veya ikili kod oluŐturulması ve daha sonra yeniden oluŐturulmuŐ makine kodunun orijinal makine kodu ile karŐılaŐtırılması ile yapılır.

Biçimsel Yöntemler

Biçimsel yöntemler, yazılım ve donanım sistemlerinin geliştirilmesi ve doğrulanması için matematiksel teknikler ve araçların kullanılmasını içerir.

2.4. GÜVENLİK TEST ARAÇLARI

2.4.1. Kaynak Kod Analizi Test Araçları

Statik analiz ile uygulama çalıştırılmaksızın, uygulamanın kaynak kodları üzerinden analiz yapılmaktadır. Bu analiz gözden geçirme şeklinde olabileceği gibi otomatik araçlar aracılığıyla olabilir.

Otomatik araçlar ile hem kod kalitesi ölçülebilmekte hem de güvenlik hatalarına sebebiyet verebilecek kodlama hataları tespit edilebilmektedir. Kod kalitesi ölçümünde pep8 vb. kodlama stilleri üzerinden kontroller yapılır. Bu kontroller kodlamanın yazım biçimlerini, kod içinde bulunan boşluklar, değişken isimleri, fonksiyonların girdi sayılarını, satır sayılarını ve benzeri özellikleri kontrol eder. Otomatik araçlar aracılığıyla, güvenlik açıklıklarına sebebiyet verebilecek kodlama hataları da tespit edilebilmektedir. Örneğin boyut kontrolü yapılmadan kopyalanan bir karakter katarının uygulama için taşma hatası yaratabileceği yerler otomatik olarak tespit edilebilmektedir.

Otomatik araçlar ürettiği hata çıktılarının bir kısmı yanlış alarm (false pozitif) hatalardır. Bunlar gerçekte güvenlik açığı oluşturmadığı halde statik analiz araçları tarafından tespit edilen hatalardır. Yanlış alarm hataların tespit edilmesi için otomatik araçların çıktıları yine manuel olarak gözden geçirilmelidir.

Statik kod analizde kullanılabilir örnek araçlar aşağıda listelenmiştir.

Tablo 1 Statik Analiz Araçları

Araç Adı	Web Adresi
Boon	http://www.cs.berkeley.edu/~daw/boon
Find Security Bugs	http://h3xstream.github.io/find-sec-bugs/
FindBugs	http://findbugs.sourceforge.net
FlawFinder	http://www.dwheeler.com/flawfinder
FxCop	https://www.microsoft.com/en-us/download/details.aspx?id=6544
Google CodeSearchDiggity	http://www.bishopfox.com/resources/tools/google-hacking-diggity/attack-tools/
Oedipus	http://www.darknet.org.uk/2006/06/oedipus-open-source-web-application-security-analysis/
OWASP LAPSE	https://www.owasp.org/index.php/OWASP_LAPSE_Project
OWASP O2 Platform	https://www.owasp.org/index.php/OWASP_O2_Platform

Araç Adı	Web Adresi
Owasp Orizon	https://www.owasp.org/index.php/Category:OWASP_Orizon_Project
OWASP WAP	https://www.owasp.org/index.php/OWASP_WAP-Web_Application_Protection
phpcs-security-audit	https://github.com/Pheromone/phpcs-security-audit
PMD	http://pmd.sourceforge.net/
SonarQube	http://sonarqube.org
Splint	http://splint.org

2.4.2. Sızma Testi Araçları

Sızma testlerinde yöntem olarak kara kutu, gri kutu ve beyaz kutu yaklaşımları kullanılmaktadır. Beyaz kutu yaklaşımında kaynak kod dâhil sistem hakkında tüm bilgilere sahipken, gri kutu yaklaşımında ise kapsam hakkında sınırlı bilgi ile sızma testi gerçekleştirilir. Kara kutu yaklaşımında test edilen sistem hakkında hiçbir ön bilgi bulunmadığı kabul edilmektedir. Testi gerçekleştiren ekibin kaynak koda erişimi bulunmamaktadır. Sisteme yapılan girdiler karşısında sistemin verdiği çıktılar gözlemlenerek sistem hakkında bilgiler edinilip, elde edilen bilgiler ışığında yeni testler yapılmaktadır. Sızma testleri aşağıdaki adımları içermektedir:

- **Keşif:** Pasif ve aktif bilgi toplama aşamalarından oluşur. Pasif bilgi toplamada test edilen sistemle etkileşime girilmeden, çevrimiçi kaynaklar kullanılarak hedef sistem ile ilgili bilgi toplanır. Aktif bilgi toplama aşamasında ise test edilen sistem ile etkileşime girilerek bilgi toplamaya çalışılır. Toplanan bilgiler uygulamanın mimarisi, kullandığı teknolojiler, sürüm bilgileri, servis bilgileri, IP bilgileri, port bilgileri, yapılandırma ayarları, geliştirici bilgileri, kullanıcı verileri gibi çeşitli bilgilerden oluşur. Keşif ve sızma testlerinde kullanılacak örnek araçlar aşağıdaki tabloda listelenmiştir.

Tablo 2 Keşif ve Sızma Testi Araçları

Aracın İsmi	Web Adresi
Aircrack-ng	https://www.aircrack-ng.org/
Cain & Abel	http://www.oxid.it/cain.html
Hping	http://www.hping.org/
THC-Hydra	http://www.thc.org/thc-hydra/
Immunity Canvas	http://www.immunityinc.com/products/canvas/
John the Ripper	http://www.openwall.com/john/
Metasploit Framework	https://www.metasploit.com/
Nmap	https://nmap.org/
p0f	http://lcamtuf.coredump.cx/p0f3/#
OpenVas	http://www.openvas.org/
Ophcrack	http://ophcrack.sourceforge.net/

Process Hacker	https://processhacker.sourceforge.io/
Recon-ng	https://bitbucket.org/LaNMaSteR53/recon-ng
Scapy	http://www.secdev.org/projects/scapy/
Sqlmap	http://sqlmap.org/
Tcpdump	https://www.tcpdump.org/tcpdump_man.html
Wireshark	https://www.wireshark.org/

- **Zafiyet Tarama:** Bu aşamada keşif aşamasında bulunan bilgilerden faydalanılarak uygulama bileşenlerinin bilinen açıklıkları belirlenir. Keşif aşamasında elde edilen bulgularda hata olabileceği veya uygulama bileşenlerinde yama uygulanmış olabileceği için açıklığın gerçekten var olup olmadığının tespiti için zafiyetin sömürülmeye çalışılması gerekmektedir. Zafiyet taramada kullanılacak örnek araçlar aşağıdaki tabloda listelenmiştir.

Tablo 3 Web Sızma Testi Araçları

Aracın İsmi	Web Adresi
Burp Suite	http://www.portswigger.net/Burp/
Metasploit	https://www.metasploit.com/
Nikto	https://cirt.net/nikto2
OWASP CAL9000	https://www.owasp.org/index.php/OWASP_WAP-Web_Application_Protection
OWASP Mantra	http://www.getmantra.com/
OWASP Pantera	https://www.owasp.org/index.php/Category:OWASP_Pantera_Web_Assessment_Studio_Project
OWASP WebScarab	https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project
OWASP ZAP	https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
OWASP Xenotix XSS Exploit Framework	https://www.owasp.org/index.php/OWASP_Xenotix_XSS_Exploit_Framework
SPIKE	http://www.immunitysec.com/resources-freesoftware.shtml
W3af	http://w3af.sourceforge.net/
Vega	https://subgraph.com/vega/

2.4.3. Rastgele (Fuzz) Test Araçları

Fuzz testi ile uygulamanın girdi noktalarına beklenmedik rasgele veriler sistematik olarak gönderilerek uygulamanın ürettiği hatalar gözlemlenir.

Öncelikle hedef yazılım üzerinde testi gerçekleştirecek girdi noktalarının tespiti edilir. Uygulamamızın girdisi bir dosya formatı (docx, pdf, jpeg, ...), bir protokole ait bir mesaj (ftp, http, arp vb.), çevresel değişken (environment variable), kayıt anahtarı girdisi (register key)

ya da kullanıcı tarafından girilen bir form deęiŐkeni Őeklinde olabilir. Sonrasında girdi noktalarına gönderilecek verinin formatına uygun veri kümesi oluŐturulur, bu kümedeki veriler ya sırayla sistematik bir Őekilde ya da rasgele örnekleme yöntemiyle belli bir sayıda uygulamada kullanılır. Rastgele testlerin uygulama girdilerinin herhangi bir hataya neden olup olmadığı sistem ve uygulama iz kayıtlarından gözlemlenir. Son aŐamada hata oluŐturan girdiler tespit edilerek bulunan hataların güvenlik açısından sömürülebilir (exploitable) olup olmadığı tespit edilir. Rastgele testler için kullanılacak örnek araçlar aŐaęıdaki tabloda listelenmiŐtir.

Tablo 4 Rastgele (Fuzz) Test Araçları

Araç Adı	Web Adresi
OWASP WSFuzzer	https://www.owasp.org/index.php/Category:OWASP_WSFuzzer_Project
Wfuzz	http://www.darknet.org.uk/2007/07/wfuzz-a-tool-for-bruteforcingfuzzing-web-applications/

2.5. GÜVENLİ YAZILIM GELİŐTİRME OLGUNLUK MODELLERİ

Güvenli yazılım geliştirme olgunluk modelleri bir kuruluşun yazılım güvenliđi pratiklerini ne derecede uyguladığını ortaya koyan derecelendirmeye dayalı ölçüm modelleridir. OWASP tarafından ortaya konulmuş olan SAMM modeli ve Building Security In Maturity Model (BSIMM) modeli aşağıda özetlenmiştir.

2.5.1. SAMM

Yazılım Güvencesi Olgunluk Modeli (SAMM) yazılım güvenliđi için kuruluşların bir strateji oluşturması ve gerçekleştirmesi için herkese açık bir modeldir. Bir kuruluş için olgunluk seviyeleri tanımlayarak güvenlik pratiklerinin kazanılması için yol haritası sağlar. Ana amacı kuruluşun mevcut yazılım güvenliđi pratiklerini değerlendirebilmesi ve dengeli bir yazılım güvence programı oluşturulabilmesini sağlamaktır.

SAMM olgunluk modeli bileşenleri Şekil 6'da verilmiştir.



Şekil 6 SAMM Olgunluk Modeli Bileşenleri

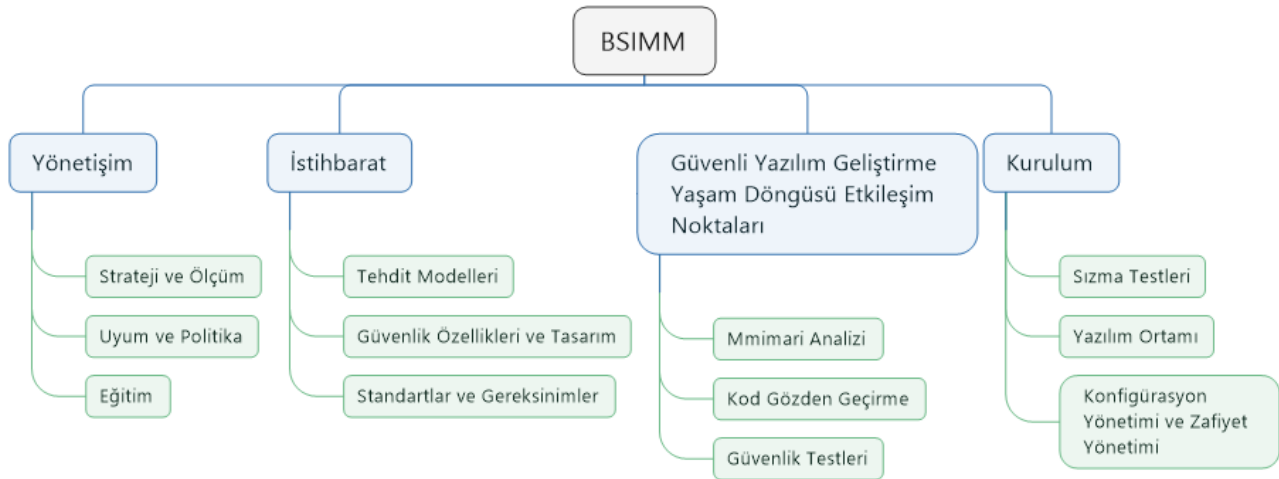
SAMM modelinde her bir güvenlik pratiđi için üç olgunluk seviyesi ve bir başlangıç seviyesi bulunmaktadır. Bu seviyelerin açıklaması aşağıdaki gibidir:

- **0. Seviye:** Bir pratiđin uygulanmadan önce kurumun durumunu belirtir.
- **1. Seviye:** Güvenlik pratiđinin ilk olarak anlaşılması ve deneme uygulamalarının yapılması durumunu belirtir.
- **2. Seviye:** Güvenlik pratiđinin etkin olarak uygulanmaya başlandıđı durumu belirtir.
- **3. Seviye:** Güvenlik pratiđinin tüm detaylarıyla uygulandıđı durumu belirtir.

2.5.2. BSIMM

BSIMM modelinde (Őekil 7) önerilen Yazılım GüvenliĐi Çerçevesi içerisinde dört ana etkinlik alanı bulunmaktadır:

- **Yönetişim:** Bir yazılım güvenliĐi inisiyatifi başlatmak, organize etmek, yönetmek ve ölçmek için gereken pratikleri içerir.
- **İstihbarat:** Güvenlikle ilgili yazılıma ilişkin bilgilerin toplanması ve işlenmesine ait pratikleri içerir. Güvenlik olaylarını önleyici yönergeler geliştirilmesi ve kurumsal tehdit modellemeyi de içeren faaliyetler tanımlar.
- **Güvenli yazılım geliştirme yaşam döngüsü etkileşim noktaları:** Yazılım geliştirme çıktıları ve süreçlerine ilişkin analiz ve güvence ile ilgili pratikleri içerir. Tüm güvenli yazılım geliştirme süreçlerinde bu ortak pratikler bulunmaktadır.
- **Kurulum:** Yazılım yapılandırma, kurulum ve bakım aşamalarında güvenlikle ilgili hususlar bu başlık altında incelenmektedir.



Őekil 7 BSIMM Etkinlik Alanları ve Faaliyetleri

BSIMM modelinde, SAMM modeline benzer şekilde üç kurumsal seviye bulunmaktadır. Pratikler ve bunların altındaki etkinlikler üç seviyeye ayrılmış şekilde tanımlanmıştır. Bir kurumun olgunluk seviyesini ölçmek için hangi etkinlikleri uyguladığı belirlenerek bir örümcek çizgesi (spider-chart) hazırlanır. Bu çizgede dünya ortalamalarına göre değerlendirme altındaki firmanın farklı alanlardaki sapması grafiksel olarak görülebilmektedir. AŐağıdaki örnekte (Őekil 8), örnek bir firmanın değerlendirme sonucu oluşturulan örümcek çizgesi yer almaktadır. Yeşil ile gösterilen çizge dünya ortalamasını, mavi ile gösterilen çizge ise örnek bir firmanın değerlendirme sonucunu göstermektedir.



Şekil 8 Örnek bir firma için BSIMM değerlendirmesi

2.6. UYGULAMA GÜVENLİĐİ

Bir uygulama, yazılım içeren bir BT çözümü olarak tanımlanabilir. Uygulama güvenliği, bir uygulama tarafından saklanan, işlenen, kullanılan ve transfer edilen kritik verilerin organizasyon tarafından istenilen biçimde korunmasını sağlar. Bu koruma sadece verilerin erişilebilirliğini, bütünlüğünü ve güvenilirliğini değil aynı zamanda da doğrulanabilirliğini ve inkâr edilememesini sağlar. Verilerin kritiklik seviyesi organizasyonun güvenlik risk değerlendirmesi sonucunda belirlenir. Koruma gerektiren kritik verinin tam olarak korunabilmesi için, aynı zamanda uygulama kaynak kodu, ikili kodu ve çalışma zamanı kodunun da korunması gerekmektedir. Aşağıdaki tabloda, bir uygulamanın kapsamını ve uygulama güvenliğinin kapsamı belirtilmiştir.

Tablo 5 Uygulama Kapsamı ve Uygulama Güvenliği Kapsamı

Bileşen	Uygulama Kapsamı	Uygulama Güvenliği Kapsamı
Kurum ve Kullanıcı Verisi	✘	✔
Uygulama Verisi	✔	✔
Roller ve Yetkiler	✔	✔
Uygulama Yönergeleri	✔	✔
Teknolojik Bağlam	✘	✔
Uygulama İle İlgili Süreçler	✘	✔
Uygulama Yaşam Döngüsü Süreci	✘	✔
İş Bağlamı	✘	✔
Yasal Bağlam	✘	✔

Uygulama güvenliđi kapsamı ve ISO 27034 Uygulama Güvenliđi standardı dikkate alınarak EK-2’de Uygulama Güvenliđi Kuralları (UGK) tanımlanmıŐtır. UGK basitçe uygulamadaki bir güvenlik zayıflıđını gidermek için kullanılan bir kontrol olarak tanımlanabilir. Örneđin, “SQL sorgularının parametrik olarak yapılması” genel bir uygulama zafiyeti olan “SQL enjeksiyonu” saldırılarını engellemek için kullanılan bir uygulama güvenliđi kuralıdır. Her bir UGK bir uygulama ve uygulamaya bađlı bađlam (context) için önem arz etmektedir. Örneđin önceki örnekte verilen “SQL sorgularının parametrik olarak yapılması” uygulama güvenliđi kuralı, veritabanları için önemlidir ve teknolojik bađlamı hedefler. Diđer bir UGK iŐ bađlamı için önemli olabilir. Örneđin, bir banka uygulamasında, bir kullanıcının diđer bir kullanıcının kullanıcı bilgilerini görmesini engelleme kontrolü iŐ bađlamı için önemlidir.

Her uygulama aynı seviyede güvenlik kontrolüne ihtiyaç duymaz. Sadece kurum içinde kullanılan hassas bilgi içermeyen bir uygulama ile internete açık kullanıcı bilgilerini içeren bankacılık uygulaması farklı güvenlik kontrollerine ihtiyaç duyabilir. Her bir UGK bir veya birden fazla güvenlik seviyesine sahip olabilir. Bu kılavuzda 3 kademeli güvenlik seviyesi tanımlanmıŐtır. Tanımlanan her bir UGK birden fazla uygulama güvenlik seviyesinde geçerli olabilir. UGK’ların hangi seviyede geçerli olduđu kural tanımlanırken belirtilmiŐtir.

- Seviye 1’de uygulanan UGK’lar sadece en temel risklere karŐı alınması gereken önlemleri tarif eder. Bu UGK’lar tüm uygulamalar için geçerlidir.
- Seviye 2’de bulunan UGK’lar, farklı endüstrilerde temel seviyenin ötesindeki tehditleri karŐılamak ve e-devlet uygulamalarında kullanılmak üzere geçerlidir.
- Seviye 3’teki UGK’lar ise uygulamanın güvenli olması için bilinen tüm yazılım güvenliđi risklerine karŐı alınması gereken önlemleri tarif eder. Bu seviyede bulunan UGK’lar insan sađlıđı, kurumsal varlıđı tehdit eden tehditlerin var olduđu bir bađlamda çalışan veya milli güvenlikle ilgili uygulamalar ve kritik altyapılar için geçerlidir.

Bir uygulamanın güvenli olduđunun söylenebilmesi için geliŐtirme sonrası gerçekte güvenlik seviyesinin, gerçekte öncesinde hedeflenen güvenlik seviyesinde büyük veya ona eŐit olması gerekir.

Farklı sektörlerde farklı bilgi ve teknoloji varlıkları mevcuttur ve farklı yasal zorunluluk ve yükümlölükler söz konusudur. Kurumların her bir uygulama için uygulamanın güvenlik seviyesini kendi ihtiyaçları ve beklentileri dođrultusunda belirlemesi ve uygulamaların bu güvenlik seviyelerini sađladığını dođrulamaları gerekmektedir. AŐađıdaki tablo, tanımlanmıŐ

uygulama güvenlik seviyelerinin farklı sektörler ve uygulamalar için nasıl yorumlanabileceğini göstermektedir.

Tablo 6 Sektörlere Göre Uygulamaların Güvenlik Seviyelerinin Sınıflandırılması

Sektör / Alan	Tehdit Profili	1. Seviye	2. Seviye	3. Seviye
Finans ve Sigorta	Sosyal mühendislik, iç tehdit, finansal siber suçlular	Tüm uygulamalar	Kişisel veri işleyen, finansal işlem bilgisi gibi hassas veri işleyen uygulamalar	Büyük miktarda hassas bilgiyi işleyen veya büyük miktarda finansal işlem yapan uygulamalar
Üretim, Lojistik, Teknoloji Altyapısı, Altyapı	Siber savaşçı, iç tehdit, siber terörist, Kötü niyetli operatör	Tüm uygulamalar	Çalışanlarla ilgili sosyal mühendislik saldırısı yapılabilecek bilgi, IP ve ticari sırlar içeren uygulamalar	Gizlilik dereceli bilgi işleyen ve depolayan, önemli ticari sırları işleyen ve depolayan uygulamalar, insan hayatına etki eden ve diğer kritik işlevleri yürüten uygulamalar
Sağlık	Sosyal mühendislik, siber savaşçı, iç tehdit, siber terörist, hacktivist, casusluk	Tüm uygulamalar	Kişisel veri işleyen, sağlık bilgisi ve ödeme bilgisi işleyen uygulamalar.	Tıbbi cihazları kontrol eden, insan sağlığını tehlikeye atabilecek bilgileri işleyen, depolayan uygulamalar. Yüksek oranda ödeme bilgisi işleyen uç terminaller.
Mağazacılık, e-Ticaret, Turizm, Gıda	Sosyal mühendislik, siber savaşçı iç tehdit	Tüm uygulamalar	Ticari ürün bilgisi, dahili firma bilgileri ve kullanıcı bilgileri işleyen ve depolayan uygulamalar	Sahtekarlığa neden olabilecek yüksek oranda ödeme bilgisi işleyen uç terminaller
e-Devlet	Siber terör, casusluk, hacktivist	Uygulanmaz	Tüm e-Devlet uygulamaları	Gizlilik dereceli bilgi işleyen, büyük miktarda kişisel veri ve kimlik bilgisi işleyen, depolayan e-Devlet uygulamaları
Kritik Altyapı (Su, Enerji, Ulaşım)	Büyük Devlet ve Organizasyonlar, Siber Terör, hacktivist	Uygulanmaz	Uygulanmaz	Tüm kritik altyapı uygulamaları
Savunma/Milli Güvenlik	Büyük Devlet ve Organizasyonlar, APT, Casusluk, Siber Terör	Uygulanmaz	Uygulanmaz	Tüm savunma/milli güvenlik uygulamaları

KAYNAKÇA

- Barnat, J., Brim, L., Ročkai, P. (2007). Scalable Multi-core LTL Model-Checking. Model Checking Software, Vol. 4595, Lecture Notes in Computer Science 187-203
- Barnett, M., Chang, B. Y. E., DeLine, R., Jacobs, B., & Leino, K. R. M. (2005). Boogie: A modular reusable verifier for object-oriented programs. In International Symposium on Formal Methods for Components and Objects (pp. 364-387). Springer Berlin Heidelberg
- Eclipse Codan, <https://wiki.eclipse.org/CDT/designs/StaticAnalysis>, EriŐim tarihi: 7 Mart 2017
- Certified Compiler, <http://compcert.inria.fr/>. EriŐim tarihi: 8 Őubat 2017
- Docker, <https://www.docker.com/>. EriŐim tarihi: 8 Őubat 2017
- Doyle, R., “Formal Methods, including Model-Based Verification and Correct-By-Construction,” (2016) Dramatically Reducing Security Vulnerabilities sessions, Software and Supply Chain Assurance (SSCA) Working Group
- ENISA Threat Landscape 2015, EriŐim tarihi: 7 Mart 2017
- Martin Fowler, “Microservices: a definition of this new architectural term,” 25 Mart 2014, <http://martinfowler.com/articles/microservices.html>, EriŐim tarihi: 7 Mart 2017
- Goertzel, K.M., Winograd, T., McKinley, H.L., Oh, L., Colon, M., McGibbon, T., Fedchak, E., Vienneau, R., Software Security Assurance State-of-the-Art Report (SOAR), Information Assurance Technology Analysis Center (IATAC) & Data and Analysis Center for Software (DACS), 2007
- The power of 10: Rules for developing safety-critical code. Computer, 39(6), 95-99
- ISO/IEC 27034-1:2011: Information technology -- Security techniques -- Application security
- Long, F., Mohindra, D., Seacord, R. C., Sutherland, D. F., & Svoboda, D. (2013). Java coding guidelines: 75 recommendations for reliable and secure programs. Addison-Wesley
- Kroening, D., Groce, A., & Clarke, E. (2004). Counterexample guided abstraction refinement via program execution. In International Conference on Formal Engineering Methods (pp. 224-238). Springer Berlin Heidelberg
- Leroy, X., “Formal certification of a compiler back-end or: programming a compiler with a proof assistant,” Proc. 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, sf. 42-54
- MISRA Consortium, Guidelines for the Use of the C Language in Critical Systems, ISBN 978-1-906400-10-1, 2013, www.misra.org.uk, EriŐim tarihi: 7 Mart 2017
- OWASP Enterprise Security API, <https://www.owasp.org>, EriŐim tarihi: 8 Őubat 2017
- Jerome H. Saltzer and Michael D. Shroeder, “The Protection of Information in Computer systems,” Proc. IEEE Vol. 63, Issue 9, September 1975, pp. 1278-1308
- SEI CERT Coding Standards, <https://www.securecoding.cert.org/> . EriŐim tarihi: 8 Őubat 2017
- Secure Coding in C and C++, 2nd ed.. Pearson Education
- Woodcock, J., Larsen, P.G., Bicarregui, J. & Fitzgerald, J., “Formal Methods: Practice and Experience,” ACM Computing Surveys, Vol. 41, Issue 4, Ekim 2009, pp. 1-36

EK 1: GÜVENLİ YAZILIM GELİŐTİRME DENETİM LİSTESİ

Güvenli yazılım geliştirme sürecinin bir proje veya kurumda uygulanmasına ilişkin denetim listesi aŐağıda belirtilmiŐtir. Bu liste ayrıca, i veya dıŐ güvenli yazılım geliştirme denetimlerinde süreç uygunluđunu kontrol etmek amacıyla kullanılabilir. Güvenli yazılım geliştirme kılavuz dokümanında belirtilen hususlar dikkate alınarak bu denetim listesi genişletilebilir veya ihtiyaçlar dođrultusunda güncellenebilir.

Denetim Maddesi	Güvenlik Seviyesi (1-2-3)	Tespit/test yöntemi
Uygulamanın üst seviye mimarisi tanımlanmalıdır.	1,2,3	Gözden geçirme
Güvenli Yazılım GeliŐtirme Kılavuzunda belirtilmiŐ olan güvenli yazılım ilkelerine uygun olarak uygulama mimari geliştirme alıŐmaları yürütülmelidir.	1,2,3	Gözden geçirme
Mimarideki tüm yazılım bileŐenleri tanımlı olmalı ve ihtiyaç duyulmayan bileŐenler kaldırılmalıdır.	1,2,3	Gözden geçirme
Risk analiziyle kullanılan teknoloji güvenlik aısından deđerlendirilmelidir.		Gözden geçirme
Tüm uygulama bileŐenlerinin işlevsel özellikleri ve güvenlik özellikleri tanımlı olmalıdır.	1,2,3	Gözden geçirme
Uygulamanın parası olmadığı halde uygulama tarafından ihtiyaç duyulan kütüphane, modül ve harici yazılımlar tanımlanmalıdır.	1,2,3	Gözden geçirme
Rafta hazır yazılım bileŐenleri güvenli yazılım depolarından temin edilmelidir.		Gözden geçirme
Uygulamanın bir tehdit modeli oluşturulmalı, riskleri tanımlanmış ve nasıl karşılanacağı belirtilmiş olmalıdır.	1,2,3	Gözden geçirme
Tehdit modelleri her bir uygulama sürümü için belgelenmiş ve gözden geçirilmiş olmalı, tasarım ve işlevsellik deđişiklikleri veya yeni tehdit ortaya ıkması durumunda güncellenmelidir.	1,2,3	Gözden geçirme
Bir Yazılım Konfigürasyon Yönetimi Planı (KYM) oluşturularak, yazılımı geliŐtiren kuruluş tarafından uygulanacak yapılandırma kontrolü ve deđişiklik yönetimi süreçleri tanımlanmalıdır. Bu süreçlere ilişkin rol ve sorumluluklar belirli olmalıdır.	1,2,3	Gözden geçirme
Konfigürasyon Yönetimi yazılımı kullanılmalı ve Konfigürasyon Yönetimi deposunun yamaları zamanında yapılmalıdır.	2,3	Gözden geçirme
Konfigürasyon Yönetimi deposunun erişim hakları periyodik olarak gözden geçirilmelidir.	2,3	Gözden geçirme
Bir Konfigürasyon Kontrol Kurulu oluşturularak uygulamanın yapılandırılmasına ilişkin deđişikliklerin yönetimi, fiziksel ve mantıksal denetimi sağlanmalıdır.	2,3	Gözden geçirme
Bir felaket kurtarma ve iş sürekliliđi planı oluşturularak uygulamanın erişilebilirlik gereksinimlerine göre işletilmesi sağlanmalıdır.	2,3	Gözden geçirme
Uygulama saldırı uğradığında izlenecek saldırı karşılık planı oluşturulmalıdır.	2,3	Gözden geçirme
Güvenlik açıklarının nasıl çözüleceđi veya ele alınacağı belirlenmelidir.	1,2,3	Gözden geçirme

Denetim Maddesi	Güvenlik Seviyesi (1-2-3)	Tespit/test yöntemi
Proje ve program yönetimi, yazılım geliştirme, tasarım, test sorumlularının kendi sorumlulukları çerçevesindeki güvenlik hususlarına ilişkin güvenli yazılım geliştirme eğitimlerini almış olmalıdır.	1,2,3	Gözden geçirme
Uygulama üzerindeki deęişiklikler Konfigürasyon Kontrol Kurulu tarafından izlenmelidir.	2,3	Gözden geçirme
Uygulama gizlilik dereceli bilgi işliyor veya içeriyorsa, veri öğeleri ve onların güvenlik sınıflandırmalarını tanımlayan bir Güvenlik Sınıflandırma Kılavuzu hazırlanmalıdır.	3	Gözden geçirme
Yazılımın güvenlik gereksinimleri tanımlanmış olmalıdır.	1, 2,3	Gözden geçirme
Güvenlik gereksinimlerinin işlevsel ve işlevsel olmayan yazılım gereksinimlerine izlenebilirliği sağlanmalıdır.	2,3	Gözden geçirme
Akış diyagramlarında ve tasarım dokümanlarında yazılımın kilitlenmesi ve yinelemeli çağrı oluşması gibi uç durumların nasıl ele alındığı tanımlanmış olmalıdır.	3	Statik analiz, kaynak kodu kalite analizi aracı
Teknik olarak güvenli ve doğrulanabilir şekilde uygulamanın ve verilerin kurtarılmasına olanak sağlayacak yazılım özellikleri mevcut olmalıdır.	2,3	Gözden geçirme
Tasarım ve kodlamadan kaynaklanan zafiyetleri tespit etmek ve azaltmak amacıyla, uygulama üzerinde kaynak kodu gözden geçirme uygulanmalı ve/veya geliştirme ortamına bütünleştirilmiş kod kalitesi araçlarıyla desteklenmelidir.	2,3	Kaynak kod gözden geçirme, kaynak kodu kalite analizi aracı
Uygulamaların işletim ortamına kurulmadan önce kriptografik olarak özetleri kontrol edilmelidir.	3	Gözden geçirme
Uygulamayla birlikte uygulamanın sürümü ile uyumlu bir uygulama yapılandırma kılavuzu oluşturulmalı ve kurulumlar bu kılavuza göre yapılmalıdır.	1,2,3	Gözden geçirme
Yetkili yöneticiler, güvenlik ile ilgili tüm ayarların bütünlüğünü denetleyebilecek araçlara sahip olmalıdır.	3	Gözden geçirme
Yazılım üzerinde etkin olarak zayıflık testleri gerçekleştirilmelidir.	2,3	Yeniden derleme ve karşılaştırma, statik analiz araçları, zayıflık analiz araçları
Sistemin başlatılması, kapatılması ve istek dışı kapanması durumlarında yazılımın güvenli bir durumda kaldığından emin olmak için test durumları oluşturulmalı ve testler periyodik olarak çalıştırılmalıdır.	3	Gözden geçirme
Güvenlikle ilgili birim testlerine ek olarak uygulamanın her bir sürümü için kod test kapsama istatistikleri tutulmalıdır.	1,2,3	Gözden geçirme
Kod gözden geçirmede bulunan hatalar bir yazılım hata izleme sistemiyle izlenmelidir.	1,2,3	Gözden geçirme
Uygulama geliştirme takımı projede tanımlı bir kod standartları kümesine uymalıdır.	1,2,3	Kaynak kodu kalite analizi aracı, kaynak kod gözden geçirme, yazılım kalitesi gözden geçirme

Denetim Maddesi	Güvenlik Seviyesi (1-2-3)	Tespit/test yöntemi
Uygulamanın her bir sürümünün devreye alınması, sistem güncellemesi veya yama uygulanması öncesi test plan ve prosedürleri oluşturulmalı ve işletilmelidir.	2,3	Gözden geçirme
Uygulamanın saldırı durumları çıkarılmalı ve kod içerisinde her bir saldırı durumu için test prosedürleri yazılmış olmalıdır.	1,2,3	Gözden geçirme
Uygulamanın güvenlik seviyesi belirlenmiş olmalıdır.	1,2,3	Gözden geçirme
Uygulamanın hedeflenen ve gerçekleşen güvenlik seviyesi ölçülmelidir.	2,3	Gözden geçirme
Periyodik olarak veritabanları, uygulamalar ve uygulama verisi yedeklemesi yapılmalıdır.	2,3	Gözden geçirme
Uygulama kurulmadan önce uygulamanın kriptografik özetleri üretilmelidir. Kurulumlarda bu özetler kontrol edilmelidir.	1,2,3	Gözden geçirme
Yazılım kaynak kodu ve yazılımın çalıştırılabilir kodunun yedekleri, geliştirme ve işletim ortamının dışında bir konumda veya afetlerden etkilenmeyecek depolama alanı içerisinde de saklanmalıdır.	3	Gözden geçirme
Uygun güvenlik ayarları ve sürümlerine göre tüm bileşenlerin güvenlik güncellemeleri ve yamaları yapılmış olmalıdır.	1,2,3	Gözden geçirme
Şifreleme anahtarlarının nasıl yönetileceğine dair açık / belirgin bir politika belirlenmelidir.	1,2,3	Gözden geçirme
Uygulamanın iz kayıtları oluşturmak üzere güvenlikle alakalı olaylar listesi tanımlanmış olmalıdır.	1,2,3	Gözden geçirme

EK 2: UYGULAMA GÜVENLİĐİ KURALLARI

Güvenlik gereksinimleri uygulama yaşam döngüsündeki her bir aşamada tam olarak tanımlanmalı ve sürekli bir biçimde analiz edilmelidir. Buna ilaveten güvenlik ile ilgili olan gereksinimler, riskler göz önüne alınarak belirlenmiş olan sınırlamalara göre ele alınmalıdır.

Gereksinimler açık, tutarlı, tam, uygulanabilir, takip edilebilir ve doğrulanabilir olmalıdır. Bu özellikler güvenlik gereksinimleri için de geçerlidir. Bu bölümde bir yazılım projesindeki güvenlik gereksinimlerinin türetilebileceđi temel uygulama güvenliđi kuralları (UGK) ele alınacaktır. Bir UGK'nın şablonu aŐağıdaki tablodaki gibidir. Her bir kuralın bir numarası ve tanımı vardır. Ayrıca kuralın referans(lar)ı da verilmiştir. Kuralın geliştirici tarafından daha detaylı anlaşılabilmesi için detaylı açıklama yapılmıştır. Kuralın karşı koyduđu zayıflıklar ve bu zayıflıkların NIST/MITRE (<http://cwe.mitre.org/index.html>) sınıflandırmasına göre Common Weakness Enumeration (CWE) numaralandırmaları da tanımlanmıştır. Otomatik veya manuel test ile kontrol edilebilecek kurallar için test yöntemi belirtilmiştir. Bir UGK'nin hangi güvenlik seviyelerinde uygulanabileceđi bilgisi de tabloda gösterilmiştir.

Uygulama Güvenliđi Kuralı (UGK) Şablonu

Kural no				
Kural tanımı				
Referans	Güvenlik Seviyesi	1	2	3
Açıklama				
KarŐı koyduđu zayıflıklar	Zayıflık referansı			
Test yöntemi				

AŐađıda, tanımlanan uygulama güvenlik kuralları listesi verilmiŐ olup lahikalarda her uygulama güvenlik kuralları için ek bilgiler yer almaktadır.

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
1	MT-01	Uygulamanın mimarisi Güvenli Yazılım GeliŐtirme Kılavuzunda belirtilmiŐ olan güvenli yazılım ilkelerine uygun olmalıdır.			
2	MT-02	Uygulamadaki bileŐenler hata durumlarında varsayılan olarak güvenli durumlara geçmelidir.			
3	MT-03	Uygulamaya yapılan tüm eriŐim istekleri hem istek hem de yanıt zamanında yetkilendirmeye tabi tutulmalıdır.			
4	MT-05	Uygulamada ihtiyaç duyulmayan kütüphane, kod ve bileŐenler tasarımda ve uygulamada ver almamalıdır.			
5	MT-07	Geçersiz olmuŐ, potansiyel olarak tehlikeli iŐlevlerin bulunduđu kütüphane ve modüller tasarımda ve uygulamada yer almamalıdır.			
6	MT-12	Uygulama bileŐenleri birbirlerinden iyi tanımlanmıŐ güvenlik mekanizmalarıyla ayrılmalıdır. Bu bağlamda sanallaŐtırma, uygulama konteyneri, ađ ayırımı, güvenlik duvarı veya bulut tabanlı güvenlik grupları gibi mekanizmalar kullanılmalıdır.			
7	MT-13	Uygulamanın veri, iŐ mantıđı ve görüntüleme katmanları belirgin Őekilde ayrılarak her bir katmandaki güvenlik kararlarının güvenilir sistem bileŐenlerine dayanması sađlanmalıdır.			
8	KD-01	Kimlik dođrulaması, özellikle herkese açık olan sayfa ve kaynaklar dıŐındaki tüm sayfa ve kaynaklara eriŐim için önkoŐul olmalıdır.			
9	KD-02	Tüm parola alanlarında kullanıcı giriŐ yaparken kullanıcının parolası maskelenmeli ve açık olarak görünmemelidir.			
10	KD-03	Sunucu tarafında tüm kimlik dođrulama denetimleri zorunlu tutulmalıdır.			
11	KD-04	Kimlik dođrulama başarısız olduđu takdirde güvenli bir duruma geçmeli ve saldırganların yetkisiz oturum açmaları engellenmelidir.			
12	KD-05	Parola giriŐ alanları uzun ve karmaŐık bir parola girilmesini engellememeli, deyimsel parola kullanımına izin vermeli ya da teŐvik etmelidir. Örneđin, parolaların en az 15 karakter uzunluđunda girilebilmesine olanak tanınması, en az bir büyük, bir küçük harf, bir özel karakter ve bir sayı kullanılması, son 5 parolayla aynı parolanın kullanılmaması vb.			
13	KD-06	Hesaba yeniden eriŐebilecek tüm hesap kimlik dođrulama iŐlevleri (profil güncelleme, parolamı unuttum, devre dıŐı/kayıp simge, süresi dolmuŐ parola güncelleme, yardım masası vb.) en az ana kimlik dođrulama mekanizması kadar saldırılara dayanıklı olmalıdır.			
14	KD-07	DeđiŐen parola iŐlevselliđi eski parolayı, yeni parolayı ve bir parola onayını kapsamalıdır.			
15	KD-08	Tüm Őüpheli kimlik dođrulama kararları için özet veri içerecek Őekilde iz kaydı oluŐturulmalıdır.			
16	KD-09	Hesaplara iliŐkin parolaları korumak için yeterince güçlü kriptografik yöntemler (Őifreleme, özet alma) kullanılmalı ve bu kriptografik yöntemlerin kaba kuvvet saldırılarına karŐı güçlü olmalıdır.			
17	KD-10	Kimlik bilgileri uygun Őifreli bir bağlantı kullanılarak iletilmeli ve kimlik bilgilerinin girilmesi için kullanıcıya gereken tüm sayfalar / iŐlevler Őifreli bir bağlantı kullanılarak yapılmalıdır.			
18	KD-11	Unutulan parola iŐlevi ve diđer kurtarma yolları geçerli parolayı açığa çıkarmamalı ve yeni parola kullanıcıya düz metin olarak gönderilmemelidir.			
19	KD-12	Oturum açma, parola sınırlama ya da hesap unutmaya gibi iŐlevler sıralı denemelerle bilgi edinmeye olanak vermemelidir.			

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
20	KD-13	Yazılım altyapısında ya da herhangi bir bileŐen için kullanılan teknolojiye üzerinde varsayılan parolalar yer almamalıdır.			
21	KD-14	Kaba kuvvet saldırıları ya da servis dıŐı bırakma saldırıları gibi otomatik yapılan yaygın kimlik doğrulama saldırılarını önlemek için istekler azaltılmalıdır.			
22	KD-15	Uygulamanın dıŐardaki hizmetlere erişmek için kullanılan tüm kimlik doğrulama bilgileri şifrenmeli ve korunan bir yerde depolanmalıdır.			
23	KD-16	Unutulan parola ve diŐer kurtarma yolları kısa ileti, e-posta onayı, mobil onay, çevrimdıŐı onay vb. yöntemleri kullanmalıdır.			
24	KD-17	Hesaplar geçici veya kalıcı olarak kilitlenebilmelidir. Kalıcı olarak kilitlenen hesaplar eŐer üzerindeki geçici kilit kaldırılrsa da kilitle kalmalıdır.			
25	KD-18	Kimlik doğrulama için bilgi sorgulayan sorular (gizli sorular) yeterince güvenli olmadığından kullanılmamalıdır.			
26	KD-19	Hassas işlevler gerçekleştirilmeden önce, yeniden kimlik doğrulama, daha güçlü bir mekanizmayla kimlik doğrulama, ikinci faktör veya işlem imzalama gibi yöntemler uygulanmalıdır.			
27	KD-20	Kimlik doğrulama işlemlerinin başarılı olup olmadığı yanıt süresinden anlaşılmalıdır.			
28	KD-21	Kaynak kodunda veya kaynak kodu depolarında gizli bilgiler, API anahtarları ve parolalar mevcut olmamalıdır.			
29	KD-22	Kullanıcılara kullanıcı adı ve parola ya da ikinci faktörle kimlik doğrulamaya ek olarak daha güçlü bir mekanizmayla kimlik doğrulama yapabilmeleri için seçenek sağlanmalıdır.			
30	KD-23	Uygulamanın yönetim arayüzlerine güvenilmeyen taraflarca erişilmesi engellenmelidir.			
31	KD-24	Uygulama kullanıcının son başarılı oturum açma tarih ve saatini görüntülemelidir.			
32	KD-25	Uygulama kullanıcı hesaplarının yönetimini sağlayan arayüzlere sahip olmalı ve yalnızca yetkili kullanıcıların erişebilmesi sağlanmalıdır. Uygulama etkin olmayan, durdurulmuş ve sonlandırılmış hesapları raporlayabilmeli ve kaldırılmasına olanak sağlamalıdır.			
33	KD-26	Her bir kullanıcı veya kullanıcının yerine işlem yapan yazılımsal süreçler tekil olarak tanımlanabilmelidir.			
34	KD-28	Hesaplara erişim için yeniden oynatma (replay) saldırılarına dayanıklı bir kimlik doğrulama mekanizması kullanılmalıdır.			
35	KD-29	Parolalar için bir en uzun geçerlilik süresi tanımlanmış olmalıdır. ÖrneŐin Seviye 3 güvenlik için 30 gün, Seviye 1 ve 2 güvenlik için 60 gün			
36	KD-30	Açık anahtar altyapısı tabanlı kimlik doğrulama kullanılıyorsa sertifika yolu doğrulanmalıdır.			
37	KD-31	Açık anahtar altyapısı tabanlı kimlik doğrulama kullanılıyorsa özel anahtara sadece yetkili kullanıcının erişimine izin verecek mekanizmalar mevcut olmalıdır.			
38	KD-32	Açık anahtar altyapısı tabanlı kimlik doğrulama kullanılıyorsa kullanıcının sertifikası sistem üzerindeki geçerli kullanıcı veya grup bilgisi ile eşleştirilmelidir.			
39	DK-01	Uygulama, ayar ve denetim dosyaları kullanıcı verisiyle aynı konumda depolanmamalıdır.			
40	DK-03	Uygulama, bilgi ve kaynaklara yapılan mantıksal erişim için erişim denetimi politikalarına uygun olacak şekilde yetkilendirme onayını zorunlu tutmalıdır.			
41	DK-04	Uygulama, paylaşılan kaynaklar üzerinden yapılan istenmeyen bilgi akıŐlarını engellemelidir.			

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
42	DK-05	URL yeniden yönlendirmelerinin sadece bilinen "beyaz liste" adreslerine yapılması, bilinmeyen adreslere yönlendirme gerekiyorsa kullanıcının uyarılarak onayının alınması sağlanmalıdır.			
43	DK-06	Uygulamaya gönderilen veri dosyaları güvenlik açığı oluşturabilecek sızıntılara karşı kontrol edilmeli ve sonrasında giriş/çıkış komutlarına gönderilmelidir.			
44	DK-07	Güvenilmeyen kaynaklardan alınan dosyaların türü doğrulanmalı ve zararlı bir içeriğe sahip olup olmadığı kontrol edilmelidir.			
45	DK-08	Güvenilmeyen verinin dinamik olarak yüklenerek çalışan koda dahil edilmesi engellenmelidir.			
46	DK-09	Karşı alanlar arası kaynak paylaşımında (Cross-domain Resource Sharing, CORS) güvenilmeyen veri kullanılmamalıdır.			
47	DK-10	Güvenilmeyen kaynaklardan alınan dosyalar, kısıtlı izinlerle uygulama ana dizini dışında depolanmalıdır.			
48	DK-11	Web veya uygulama sunucularının, kendi sınırları dışında bulunan kaynak ve sistemlere uzak bağlantı ve erişimi varsayılan olarak engellenmelidir.			
49	DK-12	Uygulama, güvenilmeyen kaynaklardan alınmış veriyi çalıştırılabilir kod olarak koşturmamalıdır.			
50	DK-13	Desteklenmeyen, güvensiz veya teknolojisi zaman aşımına uğramış istemci teknolojileri kullanılmamalıdır.			
51	OY-01	Kullanıcı oturumu kapattığında tüm oturumlar geçersiz hale getirilebilmelidir.			
52	OY-02	Oturumlar belirli bir süre etkinlik olmadığında kendiliğinden sonlanmalıdır.			
53	OY-03	Kimlik doğrulamayla erişilen tüm sayfalardan oturum kapatma işlevine erişilebilmelidir.			
54	OY-04	Oturum kimliğinin URL, hata mesajları ve iz kayıtları içerisinde yer almaması sağlanmalıdır. URL içerisinde oturum kimliğinin yeniden yazılması engellenmelidir.			
55	OY-05	Tüm kimlik doğrulama ve yeniden kimlik doğrulama işlemleri sonucunda yeni bir oturum ve yeni bir oturum kimliği üretilmelidir.			
56	OY-06	Yalnızca uygulama tarafından üretilen oturum kimliklerinin uygulamada aktif oturum kimliği olarak kullanıldığı doğrulanmalıdır.			
57	OY-07	Oturum kimlikleri yeterince uzun olmalı, rastgele olmalı ve etkin oturumlar içerisinde tekil olmalıdır.			
58	OY-08	Uygulama tarafından etkin ve aynı zamanlı oturumların sayısı sınırlandırılabilir.			
59	OY-09	Her bir kullanıcının uygulamadaki etkin oturumları görüntülenebilmeli, kullanıcı herhangi bir etkin oturumunu sonlandırabilmelidir.			
60	OY-10	Parola değişimi işlemi sonrasında kullanıcıya tüm etkin oturumları sonlandırma seçeneği sunulmalıdır.			
61	OY-11	Oturum sonlandığında oturum ile ilgili tüm geçici depolama alanları ve çerezler uygulama tarafından silinmelidir.			
62	OY-12	Web uygulamalarında oturum çerezlerinde HTTPOnly bayrağı etkin olmalıdır.			
63	OY-13	Uygulama her ürettiği oturum kimliğini yalnızca bir kez kullanmalıdır.			
64	OY-14	Tanımlama bilgilerinde depolanan oturum kimliklerinin yolları, uygulama için uygun kısıtlayıcı bir değere ayarlanmalı ve kimlik doğrulama oturumu belirteçleri ayrıca "HttpOnly" ve "secure" olarak yapılandırılmalıdır.			
65	ED-01	Kullanıcı sadece yetkilendirildiği uygulama bileşenlerine ve kaynaklara erişebilmeli ve bunları kullanabilmelidir.			
66	ED-02	Uygulama bir kullanıcının diğer kullanıcılara ait hassas bilgilere erişimini engellemelidir.			

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
67	ED-03	Uygulama izinlerde dolaşıma gerekli haller dışında izin vermemelidir. Var olan dosyalara veya dosyalar hakkındaki üst verilere erişilememedir.			
68	ED-05	Uygulama, fonksiyonlara, kaynaklara ya da verilere erişim sürelerini, kullanım oranlarını veya kullanım sıklığını değiştirebilmelidir.			
69	ED-07	Uygulama içerik duyarlı erişim denetimi yapabilmelidir (zaman, konum, öznitelikler vb.).			
70	ED-08	Uygulama verilere ve kaynaklara erişim için, erişim denetimi politikalarına uygun şekilde yetkilendirmeyi sağlamalıdır.			
71	ED-09	Uygulama veri akışı kontrol politikalarına dayalı olarak sistem içi veri akışı için erişim denetimi yapılmalıdır.			
72	ED-10	Uygulama veri akışı kontrol politikalarına dayalı olarak sistemler arası veri akışı için erişim denetimi yapılmalıdır.			
73	ED-11	Uygulama yetkisiz kullanıcıların, güvenlikle ilişkili işlevleri kullanmasına, güvenlik önlemlerini değiştirmesine, kapatmasına, atlatmasına izin vermemelidir.			
74	ED-12	Uygulama denetim kayıtlarının yetkisiz okunmasını, değiştirilmesini ve silinmesini engellemelidir.			
75	ED-13	Uygulama, kurumsal bilgi sistemlerinde saklanan ve kendi sorumluluğunda olmayan verilerin değiştirilebilmesini engellemelidir.			
76	GK-01	Uygulama, güvenlik güncellemeleri ve yamaları yapılmış bileşenlerden oluşmalıdır.			
77	GK-02	Uygulama sunucuları ve veritabanı sunucuları gibi bileşenlerin arasındaki iletişim şifreli olmalıdır.			
78	GK-03	Uygulama sunucuları ve veritabanı sunucuları gibi bileşenlerin arasındaki iletişimde ihtiyaç duyulan en az yetki sahip hesaplar kullanılmalıdır.			
79	GK-04	Uygulama kurulumları, yeterince korunaklı, korumalı ve ayrıştırılmış şekilde yapılmalıdır.			
80	GK-08	Sistem seviyesinde erişimi olan diller ile geliştirilmiş uygulamalar ASLR, DEP ve güvenlik denetimleri gibi tüm güvenlik bayrakları etkin olacak şekilde derlenmiş olmalıdır.			
81	GK-09	Uygulama, yapılandırma değişiklikleri ile ilgili erişimleri kısıtlamalı ve yapılandırma değişiklikleri için iz kayıtları oluşturmalıdır.			
82	GK-11	Uygulamada kurulum ile gelen gereksiz hesaplar olmamalıdır.			
83	GK-14	Uygulama, sistem kaynakların azalması durumunda bir yöneticiye uyarı vermelidir.			
84	GK-15	Uygulama, güncelleme bildirimlerini ya da güvenlik uyarılarını e-posta, SMS veya alternatif iletişim kanallarıyla iletebilmelidir.			
85	GK-16	Alternatif ve güvenli olmayan erişim yolları ile uygulamaya erişilememelidir.			
86	GK-17	Uygulama, başarısız sistem başlatma, başarısız sonlandırma veya başarısız kapatma gibi işlemlerde güvenli bir duruma geçmelidir.			
87	KM-01	Tüm kriptografik modüllerin, güvenli bir şekilde hataya düőtüğü doğrulanmalıdır. Hata yönetimi "Oracle Padding" atağına imkan tanımayacak şekilde olmalıdır.			
88	KM-02	Tüm rastgele üretilen sayılar, dosya isimleri, global eşsiz değerler (GUID) ve karakter dizilerinin saldırgan için tahmin edilemez olması sağlanmalıdır. Rastgele sayıların yüksek entropiye sahip olarak üretilmelidir.			
89	KM-06	Kriptografik servisleri kullanan bileşenlerin anahtarlara erişimi olmamalıdır. Kriptografik prosesler ve ana anahtarlar izole edilmelidir.			
90	KM-07	Tüm anahtar ve şifreler kullanımları tamamlandığında, tamamen sıfırlanarak yok edilmelidir.			

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
91	KM-08	Tüm anahtar ve parolalar deęiŐtirilebilir olmalıdır ve kurulum esnasında oluŐturulmalı veya deęiŐtirilmelidir.			
92	KM-13	Uygulamada Őifreleme, anahtar deęiŐimini, dijital imzalama veya özet alma gibi fonksiyonlar bulunuyorsa TS ISO/IEC 19790-24759 onaylı kriptografik modüller ve rasgele sayı üretetçeri kullanılmalıdır.			
93	KM-14	Uygulama anahtar deęiŐimini yapacaksa, anahtar deęiŐimini için güvenli kanal oluŐturulmadan önce uç noktalar için kimlik denetimi yapılmalıdır ve Őifreleme yapılmalıdır.			
94	KM-15	Uygulama saklanan kurumsal verilerin yetkisiz bir Őekilde deęiŐtirilmesini engellemek için kriptografik yöntemler uygulamalıdır.			
95	VK-01	Tüm formlarda istemci tarafında yapılan ön bellekleme işlevsellięi önemli veriler için kapatılmalıdır.			
96	VK-02	Uygulama-istemci arasındaki hassas veri iletişim için HTTP baŐlığı veya gövdesi kullanılmalıdır.			
97	VK-03	Sunucu üzerinde saklanan önemli verilerin ön belleklenmiŐ ya da geçiçi üretilmiŐ kopyaları Őifreli ve güvenli bir Őekilde saklanmalıdır.			
98	VK-04	Uygulama, saklama gereksinimi sona erdikten sonra önemli verileri güvenlik sonunu yaratmayacak Őekilde silinmelidir.			
99	VK-05	Sunucuya gelen isteklerin öngörölmeyen bir sayıda ya da büyüklükte olup olmadıęı kontrol edilebilmelidir.			
100	VK-06	İz kaydı bilgilerinin erişimi sadece yetkili kullanıcılara açık tutulmalı ve güvenlik denetimi yapılmalıdır.			
101	VK-07	Bellekte tutulan önemli veriler gereksinimi sona erdięinde güvenlik ihlali oluŐturamayacak Őekilde silinmelidir.			
102	HK-01	Uygulama, hassas veri ve kişisel verileri içeren hata mesajı veya iz kaydı üretmemelidir.			
103	HK-02	Uygulamada hata durumunda erişim varsayılan olarak engellenmelidir.			
104	HK-03	Uygulama, tanımlanan güvenlik olayları listesinde tanımlanan tüm olayların başarılı ve başarısızlık durumları için iz kayıtları oluŐturabilmelidir.			
105	HK-04	İz kayıtlarında olayların zaman sıralamasına ilişkin araştırma yapılabilecek Őekilde zaman bilgisi yer almalıdır.			
106	HK-05	İz kayıtlarını okuyan araçlarda istenilmeyen bir işlemi yapacak kayıt üretilmemelidir.			
107	HK-06	Uygulama tarafından üretilen iz kayıtları hassas bilgi içermemelidir.			
108	HK-07	İz kaydı bilgileri 5651 sayılı kanuna uygun Őekilde elektronik olarak imzalanmalıdır.			
109	HK-08	Uygulama, uygulama sunucusu ele geçirildięinde iz kayıtlarının deęiŐtirilmesini veya ortadan kaldırılmasını izin vermemelidir.			
110	IG-01	Güvenilen bir sertifika otoritesinden her Transport Layer Security (TLS) sunucu sertifikasına bir güven zinciri oluŐturulabilmeli ve her sunucu sertifikası geçerli olmalıdır.			
111	IG-02	Kimlik doęrulaması yapılmıŐ, hassas veriler ya da işlevler içeren ve güvensiz ya da ŐifrelenmemiŐ protokollerle yapılan tüm baęlantılar (iç ve dıŐ) için TLS protokolünün yaygın kullanılan son sürümü üzerinden yapılmalıdır.			
112	IG-03	Uygulama iç iletişimindeki TLS baęlantı hatası durumları için iz kaydı oluŐturulmalıdır.			
113	IG-04	Hassas bilgileri ve işlevleri kapsayan dıŐ sistemlerle olan tüm baęlantılar, kimlik doęrulamasına tabi olmalıdır.			
114	IG-05	Uygulamanın TLS gerçektelemesi, onaylanmıŐ ve en güvenli çalıŐma modunda çalıŐacak Őekilde yapılandırılmıŐ olmalıdır.			
115	IG-06	HTTP açık anahtar sabitlemesi ürün ve yedekleme açık anahtarlarıyla gerçekteŐtirilmelidir (Örnek			

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
		https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning			
116	IG-07	HTTP Sert Taşıma Güvenliđi (HTTP Strict Transport Security, HSTS) üst bilgileri tüm isteklerde ve alt alanlarda yer almalıdır.			
117	IG-08	Operasyonel web sitesi URL bilgisi, web tarayıcı sağlayıcıları tarafından tutulan önceden yüklenmiş Taşımalı Güvenlik Alanları listelerinde yer almalıdır (Örneđin: https://www.chromium.org/hsts)			
118	IG-09	Uygulamada, ađı dinleyen saldırganların trafiđi kaydetmesini engellemek için ileri gizlilik şifrelemeleri kullanılmalıdır.			
119	IG-10	Uygulama, Çevrimiçi Sertifika Durum Protokolü Damgalama (OCSP stapling) gibi yöntemlerle sertifika iptal denetimi gerçekleştirebilecek şekilde yapılandırılmalıdır (https://en.wikipedia.org/wiki/OCSP_stapling).			
120	IG-11	Sertifikalarda ve sertifikanın tüm hiyerarşisinde yalnızca güçlü algoritmalar ve protokoller kullanılmalıdır.			
121	IG-12	Özellikle yaygın yapılandırmalar, şifreler ve algoritmalar güvensiz hale geldiğinden, TLS ayarları en güncel yapılandırma önerileri ile uyumlu olmalıdır (https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet).			
122	IG-13	Kullanıcı erişimi için kimlik doğrulaması gerektiren uygulamalar, kullanıcı tarafından başlatılan iletişim oturumunu sonlandırmak için bir oturum kapatma yeteneđi sağlamalıdır ve bu yetenek oturumun sonunda iletişim oturumu ile ilişkili tüm ađ bağlantılarını sona erdirmelidir.			
123	IG-15	Uygulama, kimliđi doğrulanmış iletişim oturumlarının güvenilir olarak sonlandırıldığını belirten ve kolay anlaşılabilen bir çıkış iletisi görüntülemelidir.			
124	IM-01	Uygulama iş mantığını doğru bir şekilde gerçekleştirmeli, iş mantığındaki akışlar yazılımda beklenen sırada gerçekleşmeli, gereken adımlar atlanmamalı, adımların insanların yapabileceđi süreler içinde gerçekleştirildiđi kontrol edilmeli ve çok yüksek sıklıkla gönderilen istekler tespit edilmelidir.			
125	IM-02	Uygulamada çeşitli sınırlamalar tanımlanabilmeli ve bu sınırlamalar her bir kullanıcı için uygulanabilecek şekilde tanımlanabilmelidir. Uygulama, ihlal veya olađanüstü durumlar için doğru uyarı ve otomatik tepkiler verebilmelidir.			
126	KI-01	Uygulama, harici bir program çalıştırıyorsa, organizasyon politikalarına doğrultusunda yetkisiz olarak program çalıştırılmasını ve yapılandırılmasını engellemelidir.			
127	KI-04	Uygulama hassas bilgileri formlarda bulunan gizli alanlarda saklamamalıdır.			
128	KI-05	Uygulama Cross-Site Request Forgery (CSRF)'dan kaynaklanan açıklıklardan korunma mekanizmasına sahip olmalıdır.			
129	KI-07	Uygulama veri gösterim biçimlerindeki farklılıklardan kaynaklanan açıklıklardan korunmalıdır.			
130	MG-01	Gizli anahtarlar, API token ya da şifreler mobil uygulamalarda dinamik olarak oluşturulmalıdır.			
131	MG-02	Uygulama gerekli işlev ve kaynaklar için minimum izne sahip olmalıdır.			
132	MG-03	Uygulama, aynı cihazdaki diđer uygulamalar için hassas bilgileri dışarı aktarmamalıdır.			
133	MG-04	Uygulamanın içerik sağlayıcıları, aktiviteleri tüm girdileri doğrulamalıdır.			

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
134	GC-02	Uygulamanın çalışma ortamı, bellek taşması saldırılarına dayanıklı olmalıdır veya mevcut güvenlik mekanizmaları bellek taşmasını engellemelidir.			
135	GC-03	Sunucuda yapılan girdi doğrulama hataları, isteğın reddi ile sonuçlanmalı ve iz kaydı oluşturulmalıdır.			
136	GC-04	Uygulama tarafından, istemci ve sunucu tarafında, kabul edilen her bir veri tipi için girdi doğrulama denetimi yapılmalıdır.			
137	GC-05	Bütün veritabanı sorguları, parametre olarak yapılmalı ve veritabanına erişimde kullanılan dile karşı (SQL, NoSQL vb.) enjeksiyon saldırılarını önleyebilecek denetimler yapılmalıdır.			
138	GC-06	Uygulama, yetki onaylama hizmetlerinin (LDAP, Active Directory) enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.			
139	GC-07	Uygulama, işletim sistemi komut enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.			
140	GC-08	Uygulama, girdi alınan içerik bir dosya yolu içeriyorsa, uzak ya da yakın dosya içerme açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.			
141	GC-09	Uygulama, XML açıklıklarını (XPath sorgu saldırıları, XML harici öge saldırıları, XML enjeksiyonu vb.) önleyici güvenlik denetimlerini yapmalıdır.			
142	GC-10	Eğer uygulamanın altyapısını oluşturan teknoloji otomatik çoklu parametre birleştirme yeteneğine sahipse kötü amaçlı parametre ekleme saldırılarına karşı koruma sağlanmalıdır.			
143	GC-11	Uygulama, özellikle altyapı teknolojisi istek parametrelerinin kaynağı arasında bir ayırım yapmıyorsa, HTTP parametre kirliliğı saldırılarına karşı savunma yapabilmelidir.			
144	GC-12	HTML form alanlarının veri girdileri, REST çağrıları, HTTP üst başlıkları, çerezler, toplu işlem dosyaları, RSS beslemeleri gibi veri girdileri için doğrulama denetimi yapılmalıdır.			
145	GC-14	Yapısal olmayan veriler için izin verilen karakterler ve uzunluklar, verinin içeriğinde olabilecek olası zararlı karakterlerin denetlenmelidir.			
146	SG-01	Uygulama, web servislerini iyi yapılandırılmış en az TLS v1.2 ve muadil güvenlik önlemi sunan bir protokol ile sunacak şekilde tasarlanmalıdır.			
147	SG-02	Uygulama, web servis kimlik doğrulama ve yetkilendirmesi için oturum temelli yapılar kullanacak şekilde tasarlanmalıdır.			
148	SG-03	Uygulamanın sunduğı SOAP temelli web servisleri en az (Web Services-Interoperability Basic Profile v1.0 ve üstü) ile uyumlu olacak şekilde tasarlanmalıdır.			
149	SG-04	Uygulama, web servis yapılandırma ve yönetim işlevlerine sadece yetkili kullanıcıların erişebilmesini sağlamalıdır.			
150	SG-05	Uygulama, web servis girdilerini kullanmadan önce gidilerin şekli (XML ve JSON şemalarına uygunluk, parametre beyaz listesi) uygunluğunu ve içeriğini çeşitli saldırılara karşı (XML bombalama, dış varlık saldırısı, kusurlu XML yapısı, tekrarlamalı girdi vb.) kontrol etmelidir.			
151	SG-06	Uygulama, sunucu ve istemci tarafında dil kodlaması (encoding) saldırılarına karşı dayanıklı olmalıdır.			
152	SG-07	Uygulama, web servisi ile gönderilen veride betik (script) içermeyecek şekilde tasarlanmalıdır.			
153	SG-08	Uygulama, web servislerinden şifreli olarak paylaşılan verileri yine şifreli olarak saklayacak şekilde tasarlanmalıdır.			
154	KV-01	Uygulama, kişisel veriler üzerinde işlem yapılması ana amaç olmayan durumlarda kişisel verileri maskeleyerek görüntülemeli, aktarmalı veya işlemelidir.			

GÜVENLİ YAZILIM GELİŐTİRME KILAVUZU – SÜRÜM 1.0

Sıra No	UGK No	UGK Açıklaması	Uygulanacağı Güvenlik Seviyesi		
			1	2	3
155	KV-02	Uygulama, kişisel verileri şifreli olarak saklamalı ve bu verilerin taşınmasında korumalı iletişim kanallarını kullanmalıdır.			
156	KV-03	Uygulamanın istemci tarafında çalışan kodları, kişisel verileri başka ortamlara aktarmamalı (konsola yazma, başka dosya olarak kaydetme, yerel veya uzak uygulamalara transfer etme vb.), güvensiz ortamlarda (ortak dizin, USB disk vb.) güvensiz yöntemlerle (açık metin olarak, zayıf şifreleme algoritma kullanarak şifreleme vb.) saklamamalıdır.			
157	KV-04	Kullanılan veritabanının dışarıya aktarımı ancak veritabanı yönetim yetkisi olan hesaplarla yapılmalı ve öncesinde veritabanındaki kişisel verilerin silinmesi sağlanmalıdır.			

EK 2-Lahika-01: Mimari, Tasarım ve Tehdit Modelleme

Kural no: MT-01			
Kural tanımı: Uygulamanın mimarisi Güvenli Yazılım Geliőtirme Kılavuzunda belirtilmiő olan güvenli yazılım ilkelerine uygun olmalıdır.			
Referans: GYGK/CWE	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Güvenli Tasarım İlkelerinin İhlali	Zayıflık referansı: CWE-657	3	
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme			

Kural no: MT-02			
Kural tanımı: Uygulamadaki bileőenler hata durumlarında varsayılan olarak güvenli durumlara geçmelidir.			
Referans: GYGK/CWE	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Güvensiz olarak başarısız olma (Başarısız durumda açık duruma düşme)	Zayıflık referansı: CWE-636	3	
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme			

Kural no: MT-03			
Kural tanımı: Uygulamaya yapılan tüm erişim istekleri hem istek hem de yanıt zamanında yetkilendirmeye tabi tutulmalıdır.			
Referans: GYGK/CWE	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Tüm erişimleri denetlememek	Zayıflık referansı: CWE-638	3	
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme			

Kural no: MT-05			
Kural tanımı: Uygulamada ihtiyaç duyulmayan kütüphane, kod ve bileőenler tasarımda ve uygulamada ver almamalıdır.			
Referans: OWASP ASVS 3.0.1 1.1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Zayıf kod kalitesi, ölü kod	Zayıflık referansı: CWE-398, CWE-561, CWE-637	3	
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme			

Kural no: MT-07				
Kural tanımı: Geçersiz olmuş, potansiyel olarak tehlikeli işlevlerin bulunduğu kütüphane ve modüller tasarımda ve uygulamada yer almamalıdır.				
Referans: OWASP ASVS 3.0.1 1.2	Güvenlik Seviyesi:		2	3
Karşı koyduğu zayıflıklar: Zayıf kod kalitesi, eskimiş işlevlerin kullanımı, potansiyel olarak tehlikeli işlev kullanımı	Zayıflık referansı: CWE-477, CWE-676, CWE-749			
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme				

Kural no: MT-12				
Kural tanımı: Uygulama bileşenleri birbirlerinden iyi tanımlanmış güvenlik mekanizmalarıyla ayrılmalıdır. Bu bağlamda sanallaştırma, uygulama konteyneri, ağ ayırımı, güvenlik duvarı veya bulut tabanlı güvenlik grupları gibi mekanizmalar kullanılmalıdır.				
Referans: OWASP ASVS 3.0.1 1.8	Güvenlik Seviyesi:	1	2	3
Karşı koyduğu zayıflıklar: Bölümlenimin yetersizliği	Zayıflık referansı: CWE-653			
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, gözden geçirme				

Kural no: MT-13				
Kural tanımı: Uygulamanın veri, iş mantığı ve görüntüleme katmanları belirgin şekilde ayrılarak her bir katmandaki güvenlik kararlarının güvenilir sistem bileşenlerine dayanması sağlanmalıdır.				
Referans: OWASP ASVS 3.0.1 1.9	Güvenlik Seviyesi:	1	2	3
Karşı koyduğu zayıflıklar: Koruma mekanizması hatası, güvenlik üzerine yönetici kontrolünün olmaması	Zayıflık referansı: CWE-653, CWE-654			
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, gözden geçirme				

EK 2-Lahika-02: Kimlik Doğrulama

Kural no: KD-01			
Kural tanımı: Kimlik doğrulaması, özellikle herkese açık olan sayfa ve kaynaklar dışındaki tüm sayfa ve kaynaklara erişim için önkoşul olmalıdır.			
Referans: OWASP ASVS 3.0.1 2.1	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Tüm erişimleri denetlememek	Zayıflık referansı: CWE-638	3	
Test yöntemi: Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, biçimsel yöntemler			

Kural no: KD-02			
Kural tanımı: Tüm parola alanlarında kullanıcı giriş yaparken kullanıcının parolası maskelenmeli ve açık olarak görünmemelidir.			
Referans: OWASP ASVS 3.0.1 2.2	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Parola alanında maskeleyenin eksik olması	Zayıflık referansı: CWE-549	3	
Test yöntemi: Web uygulama açıklık tarayıcısı, tasarım gözden geçirme			

Kural no: KD-03			
Kural tanımı: Sunucu tarafında tüm kimlik doğrulama denetimleri zorunlu tutulmalıdır.			
Referans: OWASP ASVS 3.0.1 2.4	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Hatalı kimlik doğrulama, yalnızca istemci tarafında kimlik doğrulama yapılması	Zayıflık referansı: CWE-287, CWE-603	3	
Test yöntemi: Gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı			

Kural no: KD-04			
Kural tanımı: Kimlik doğrulama başarısız olduđu takdirde güvenli bir duruma geçilmeli ve saldırganların yetkisiz oturum açmaları engellenmelidir.			
Referans: OWASP ASVS 3.0 2.6	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Yetersiz kimlik doğrulama	Zayıflık referansı: CWE-287	3	
Test yöntemi: Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test			

Kural no: KD-05			
Kural tanımı: Parola giriŐ alanları uzun ve karmaŐık bir parola girilmesini engellememeli, deyimsel parola kullanımına izin vermeli ya da teŐvik etmelidir. Örneđin, parolaların en az 15 karakter uzunluđunda girilebilmesine olanak tanınması, en az bir büyük, bir küçük harf, bir özel karakter ve bir sayı kullanılması, son 5 parolayla aynı parolanın kullanılmaması vb.			
Referans: OWASP ASVS 3.0.1 2.7	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Parola gereksinimlerinin yetersiz olması	Zayıflık referansı: CWE-521		3
Test yöntemi: Gözden geçirme, kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı			

Kural no: KD-06			
Kural tanımı: Hesaba yeniden erişebilecek tüm hesap kimlik doğrulama işlemleri (profil güncelleme, parolamı unuttum, devre dıŐı/kayıp simge, süresi dolmuş parola güncelleme, yardım masası vb.) en az ana kimlik doğrulama mekanizması kadar saldırılara dayanıklı olmalıdır.			
Referans: OWASP ASVS 3.0.1 2.8	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Unutulmuş parola için zayıf parola kurtarma yöntemleri kullanma	Zayıflık referansı: CWE-640		3
Test yöntemi: Tasarım gözden geçirme, kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı, biçimsel yöntemler			

Kural no: KD-07			
Kural tanımı: DeđiŐen parola fonksiyonu eski parolayı, yeni parolayı ve bir parola onayını kapsamalıdır.			
Referans: OWASP ASVS 3.0.1 2.9	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: DođrulanmamıŐ parola deđiŐimi, yetersiz kimlik doğrulama	Zayıflık referansı: CWE-620, CWE-287		3
Test yöntemi: Gözden geçirme, kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı			

Kural no: KD-08			
Kural tanımı: Tüm Őüpheli kimlik doğrulama kararları için özet veri içerecek Őekilde iz kaydı oluşturulmalıdır.			
Referans: OWASP ASVS 3.0.1 2.12	Güvenlik Seviyesi:		2
KarŐı koyduđu zayıflıklar: AŐırı kimlik doğrulama denemelerini yeterince kısıtlamama	Zayıflık referansı: CWE-307 (CWE Top-25, #21)		3
Test yöntemi: Tasarım Gözden Geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: KD-09			
Kural tanımı: Hesaplara ilişkin parolaları korumak için yeterince güçlü kriptografik yöntemler (şifreleme, özet alma) kullanılmalı ve bu kriptografik yöntemlerin kaba kuvvet saldırılarına karşı güçlü olmalıdır.			
Referans: OWASP ASVS 3.0.1 2.13	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: Kimlik bilgilerinin yetersiz korunması, yetersiz güçte şifreleme, düşük işlem gücüyle kırılabilir kriptografik özetlerin parolayı korumak amacıyla kullanılması	Zayıflık referansı:	CWE-522, CWE-326, CWE-916	
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, kaynak kodu gözden geçirme, güvenlik kütüphaneleri doğrulama, biçimsel yöntemler			

Kural no: KD-10			
Kural tanımı: Kimlik bilgileri uygun şifreli bir bağlantı kullanılarak iletilmeli ve kimlik bilgilerinin girilmesi için kullanıcıya gereken tüm sayfalar / işlevler şifreli bir bağlantı kullanılarak yapılmalıdır.			
Referans: OWASP ASVS 3.0.1 2.16	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Hassas bilgilerin şifrelenmemesi, hassas bilgilerin açık olarak iletilmesi	Zayıflık referansı:	CWE-311 (CWE Top-25, #8), CWE-319	
Test yöntemi: Kaynak kodu gözden geçirme, gözden geçirme, ağ paket dinleme, negatif test			

Kural no: KD-11			
Kural tanımı: Unutulan parola işlevi ve diğer kurtarma yolları geçerli parolayı açığa çıkarmamalı ve yeni parola kullanıcıya düz metin olarak gönderilmemelidir.			
Referans: OWASP ASVS 3.0.1 2.17	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Zayıf parola kurtarma yöntemleri kullanma, hassas bilgilerin şifrelenmemesi, hassas bilgilerin açık olarak iletilmesi	Zayıflık referansı:	CWE-640, CWE-311 (CWE Top-25, #8), CWE-319	
Test yöntemi: Gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı, ağ paket dinleme, negatif test			

Kural no: KD-12			
Kural tanımı: Oturum açma, parola sıfırlama ya da hesap unutma gibi işlevler sıralı denemelerle bilgi edinmeye olanak vermemelidir.			
Referans: OWASP ASVS 3.0.1 2.18	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Unutulmuş parola için zayıf parola kurtarma yöntemleri kullanma	Zayıflık referansı:	CWE-640	
Test yöntemi: Tasarım Gözden Geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: KD-13			
Kural tanımı: Yazılım altyapısında ya da herhangi bir bileŐen için kullanılan teknolojide üzerinde varsayılan parolalar yer almamalıdır.			
Referans: OWASP ASVS 3.0.1 2.19	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Koda gömülü kimlik bilgilerini kullanma	Zayıflık referansı: CWE-798	3	
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı, kaynak kod analiz tarayıcısı			

Kural no: KD-14			
Kural tanımı: Kaba kuvvet saldırıları ya da servis dışı bırakma saldırıları gibi otomatik yapılan yaygın kimlik doğrulama saldırılarını önlemek için istekler azaltılmalıdır.			
Referans: OWASP ASVS 3.0.1 2.20	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: AŐırı kimlik doğrulama denemelerini yeterince kısıtlamama	Zayıflık referansı: CWE-307 (CWE Top-25, #21)	3	
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: KD-15			
Kural tanımı: Uygulamanın dışardaki hizmetlere erişmek için kullanılan tüm kimlik doğrulama bilgileri şifrelenmeli ve korunan bir yerde depolanmalıdır.			
Referans: OWASP ASVS 3.0.1 2.21	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Hassas bilginin açığa çıkması, açık metin olarak veya yetersiz güvenlik korumasıyla depolanması	Zayıflık referansı: OWASP Top Ten 2013 #A6, CWE-312, CWE-922	3	
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: KD-16			
Kural tanımı: Unutulan parola ve diđer kurtarma yolları kısa ileti, e-posta onayı, mobil onay, çevrimdışı onay vb. yöntemleri kullanmalıdır.			
Referans: OWASP ASVS 3.0.1 2.22	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Unutulmuş parola için zayıf parola kurtarma yöntemleri kullanma	Zayıflık referansı: CWE-640	3	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı, biçimsel yöntemler			

Kural no: KD-17				
Kural tanımı: Hesaplar geçici veya kalıcı olarak kilitlenebilmelidir. Kalıcı olarak kilitlenen hesaplar eđer üzerindeki geçici kilit kaldırılrsa da kilitli kalmalıdır.				
Referans: OWASP ASVS 3.0.1 2.23	Güvenlik Seviyesi:		2	3
Karşı koyduđu zayıflıklar: Hesap kilitlemede geređinden fazla kısıtlayıcı yöntem kullanma	Zayıflık referansı: CWE-645			
Test yöntemi: Tasarım gözden geçirme, kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı, biçimsel yöntemler				

Kural no: KD-18				
Kural tanımı: Kimlik doğrulama için bilgi sorgulayan sorular (gizli sorular) yeterince güvenli olmadığından kullanılmamalıdır.				
Referans: OWASP ASVS 3.0.1 2.24	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Unutulmuş parola için zayıf parola kurtarma yöntemleri kullanma	Zayıflık referansı: CWE-640			
Test yöntemi: Tasarım Gözden Geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı				

Kural no: KD-19				
Kural tanımı: Hassas işlevler gerçekleştirilmeden önce, yeniden kimlik doğrulama, daha güçlü bir mekanizmayla kimlik doğrulama, ikinci faktör veya işlem imzalama gibi yöntemler uygulanmalıdır.				
Referans: OWASP ASVS 3.0.1 2.26	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Kritik bilgi ve işlevlere kimlik doğrulama olmadan erişim	Zayıflık referansı: CWE-306			
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, web uygulama açıklık tarayıcısı, biçimsel yöntemler				

Kural no: KD-20				
Kural tanımı: Kimlik doğrulama işlemlerinin başarılı olup olmadığı yanıt süresinden anlaşılamamalıdır.				
Referans: OWASP ASVS 3.0.1 2.28	Güvenlik Seviyesi:			3
Karşı koyduđu zayıflıklar: Yan zamanlama kanalı	Zayıflık referansı: CWE-385			
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, işlevsel test				

Kural no: KD-21			
Kural tanımı: Kaynak kodunda veya kaynak kodu depolarında gizli bilgiler, API anahtarları ve parolalar mevcut olmamalıdır.			
Referans: OWASP ASVS 3.0.1 2.29	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Koda gömülü kimlik bilgilerini kullanma	Zayıflık referansı: CWE-798	3	
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, Tasarım Gözden Geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: KD-22			
Kural tanımı: Kullanıcılara kullanıcı adı ve parola ya da ikinci faktörle kimlik doğrulamaya ek olarak daha güçlü bir mekanizmayla kimlik doğrulama yapabilmeleri için seçenek sağlanmalıdır.			
Referans: OWASP ASVS 3.0.1 2.31	Güvenlik Seviyesi:		3
Karşı koyduđu zayıflıklar: Tek faktörlü kimlik doğrulama kullanma	Zayıflık referansı: CWE-308		
Test yöntemi: Güvenlik kütüphaneleri doğrulama, web uygulama açıklık tarayıcısı, işlevsel test			

Kural no: KD-23			
Kural tanımı: Uygulamanın yönetim arayüzlerine güvenilmeyen taraflarca erişilmesi engellenmelidir.			
Referans: OWASP ASVS 3.0.1 2.32	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Geçersiz/hatalı yetkilendirme	Zayıflık referansı: CWE-285, CWE-863, CWE/SANS Top 25 2011 #15	3	
Test yöntemi: Güvenlik kütüphaneleri doğrulama, web uygulama açıklık tarayıcısı, işlevsel test			

Kural no: KD-24			
Kural tanımı: Uygulama kullanıcının son başarılı oturum açma tarih ve saatini görüntülemelidir.			
Referans: Appsec STIG SV-83977r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Çok sık gerçekleştirilen kimlik doğrulama denemelerini yeterince kısıtlamama	Zayıflık referansı: CWE-307, CWE/SANS Top 25 2011 #21	3	
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, Tasarım Gözden Geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: KD-25				
Kural tanımı: Uygulama kullanıcı hesaplarının yönetimini sağlayan arayüzlere sahip olmalı ve yalnızca yetkili kullanıcıların erişebilmesi sağlanmalıdır. Uygulama etkin olmayan, durdurulmuş ve sonlandırılmış hesapları raporlayabilmeli ve kaldırılmasına olanak sağlamalıdır.				
Referans: Appsec STIG SV-84913r1_rule	Güvenlik Seviyesi:		2	3
Karşı koyduğu zayıflıklar: Hatalı kullanıcı yönetimi	Zayıflık referansı: CWE-286			
Test yöntemi: Güvenlik kütüphaneleri doğrulama, web uygulama açıklık tarayıcısı, işlevsel test				

Kural no: KD-26				
Kural tanımı: Her bir kullanıcı veya kullanıcının yerine işlem yapan yazılımsal süreçler tekil olarak tanımlanabilmelidir.				
Referans: Appsec STIG SV-84149r1_rule	Güvenlik Seviyesi:	1	2	3
Karşı koyduğu zayıflıklar: Yetersiz kimlik doğrulama	Zayıflık referansı: CWE-287			
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler				

Kural no: KD-28				
Kural tanımı: Hesaplara erişim için yeniden oynatma (replay) saldırılarına dayanıklı bir kimlik doğrulama mekanizması kullanılmalıdır.				
Referans: Appsec STIG SV-84167r1_rule	Güvenlik Seviyesi:		2	3
Karşı koyduğu zayıflıklar: Kaydetme-oynatma ile kimlik doğrulamayı geçersiz kılma	Zayıflık referansı: CWE-294			
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler				

Kural no: KD-29				
Kural tanımı: Parolalar için bir en uzun geçerlilik süresi tanımlanmış olmalıdır. Örneğin Seviye 3 güvenlik için 30 gün, Seviye 1 ve 2 güvenlik için 60 gün				
Referans: Appsec STIG SV-84195r1_rule	Güvenlik Seviyesi:	1	2	3
Karşı koyduğu zayıflıklar: Parola yaşlanması uygulamama	Zayıflık referansı: CWE-262			
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, Tasarım Gözden Geçirme, kaynak kod gözden geçirme, web uygulama açıklık tarayıcısı				

Kural no: KD-30				
Kural tanımı: Açık anahtar altyapısı tabanlı kimlik doğrulama kullanılıyorsa sertifika yolu doğrulanmalıdır.				
Referans: Appsec STIG SV-84771r1_rule	Güvenlik Seviyesi:		2	3

KarŐı koyduđu zayıflıklar: Sertifikanın güven zincirini hatalı dođrulama	Zayıflık referansı: CWE-296
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler	

Kural no: KD-31			
Kural tanımı: Açık anahtar altyapısı tabanlı kimlik dođrulama kullanılıyorsa özel anahtara sadece yetkili kullanıcının erişimine izin verecek mekanizmalar mevcut olmalıdır.			
Referans: Appsec STIG SV-84773r1_rule	Güvenlik Seviyesi:	2	3
KarŐı koyduđu zayıflıklar: Anahtar yönetimi hataları	Zayıflık referansı: CWE-320		
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: KD-32			
Kural tanımı: Açık anahtar altyapısı tabanlı kimlik dođrulama kullanılıyorsa kullanıcının sertifikası sistem üzerindeki geçerli kullanıcı veya grup bilgisi ile eşleŐtirilmelidir.			
Referans: Appsec STIG SV-84775r1_rule	Güvenlik Seviyesi:	2	3
KarŐı koyduđu zayıflıklar: Sertifikanın ait olduđu bilgisayar dođrulama hatası, anahtar yönetimi hataları	Zayıflık referansı: CWE-297, CWE-320		
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

EK 2-Lahika-03: Dosyalar ve Kaynakların Güvenliđi

Kural no: DK-01			
Kural tanımı: Uygulama, ayar ve denetim dosyaları kullanıcı verisiyle aynı konumda depolamamalıdır.			
Referans: Appsec STIG SV-84931r1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Servis dışı bırakma, kötü amaçlı kod, kritik kaynak için hatalı yetkiler, hatalı varsayılan yetkiler	Zayıflık referansı:	CWE-904, CWE-730, OWASP Top Ten 2004 A9, CWE-732, CWE-275, CWE-538	
Test yöntemi: Kaynak kodu gözden geçirme, uygulama açıklık tarayıcısı			

Kural no: DK-03			
Kural tanımı: Uygulama, bilgi ve kaynaklara yapılan mantıksal erişim için erişim denetimi politikalarına uygun olacak şekilde yetkilendirme onayını zorunlu tutmalıdır.			
Referans: Appsec STIG SV-83951r1	Güvenlik Seviyesi:		3
Karşı koyduđu zayıflıklar: Geçersiz/hatalı yetkilendirme	Zayıflık referansı:	CWE-285, OWASP Top Ten 2013 A7	
Test yöntemi: Negatif test, gözden geçirme			

Kural no: DK-04			
Kural tanımı: Uygulama, paylaşılan kaynaklar üzerinden yapılan istenmeyen bilgi akışlarını engellemelidir.			
Referans: Appsec STIG SV-84857r1	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: Bilgi sızması/açıđa çıkması	Zayıflık referansı:	CWE-212, CWE-552, CWE-668	
Test yöntemi: Statik kod analizi, kaynak kodu gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: DK-05			
Kural tanımı: URL yeniden yönlendirmelerinin sadece bilinen "beyaz liste" adreslerine yapılması, bilinmeyen adreslere yönlendirme gerekiyorsa kullanıcının uyarılarak onayının alınması sağlanmalıdır.			
Referans: OWASP ASVS 3.0/3.1 16.1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Dođrulanmamış yeniden yönlendirme	Zayıflık referansı:	CWE-601, CWE-938(OWASP Top Ten 2013 A10)	
Test yöntemi: Statik kod analizi, kaynak kodu gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: DK-06			
Kural tanımı: Uygulamaya gönderilen veri dosyaları güvenlik açığı oluşturabilecek sızıntılara karşı kontrol edilmeli ve sonrasında giriş/çıkış komutlarına gönderilmelidir.			
Referans: OWASP ASVS 3.0/3.1 16.2	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Dosya yolu izleme, yerel dosya içerme, dosya çoklu ortam türü, işletim sistemi komut enjeksiyonu, bilgi sızması	Zayıflık referansı:	CWE-22, CWE-200, CWE-98, CWE-78, CWE-616	
Test yöntemi: Statik kod analizi, kaynak kodu gözden geçirme, web uygulama açıklık tarayıcısı			

Kural no: DK-07			
Kural tanımı: Güvenilmeyen kaynaklardan alınan dosyaların türü doğrulanmalı ve zararlı bir içeriđe sahip olup olmadığı kontrol edilmelidir.			
Referans: OWASP ASVS 3.0/3.1 16.3	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Kötü amaçlı yazılım	Zayıflık referansı:	CWE-434 (CWE/SANS Top 25, #9)	
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, kaynak kodu gözden geçirme			

Kural no: DK-08			
Kural tanımı: Güvenilmeyen verinin dinamik olarak yüklenerek çalışan koda dahil edilmesi engellenmelidir.			
Referans: OWASP ASVS 3.0/3.1 16.4	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Bilinmeyen kaynaktan işlevlerin yazılıma dahil edilmesi	Zayıflık referansı:	CWE-829 (CWE/SANS Top 25, #16)	
Test yöntemi: Mimari/tasarım gözden geçirme, kaynak kodu gözden geçirme, kaynak kodu zayıflık tarayıcısı			

Kural no: DK-09			
Kural tanımı: Karşı alanlar arası kaynak paylaşımında (Cross-domain Resource Sharing, CORS) güvenilmeyen veri kullanılmamalıdır.			
Referans: OWASP ASVS 3.0/3.1 16.5	Güvenlik Seviyesi:		2
Karşı koyduđu zayıflıklar: Etki alanları arası erişime gereğinden fazla izin verme	Zayıflık referansı:	CWE-942	
Test yöntemi: Tasarım Gözden Geçirme, kaynak kod gözden geçirme, işlevsel test			

Kural no: DK-10				
Kural tanımı: Güvenilmeyen kaynaklardan alınan dosyalar, kısıtlı izinlerle uygulama ana dizini dışında depolanmalıdır.				
Referans: OWASP ASVS 3.0/3.1 16.6	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Web dizini altında hassas veri içirme, güvenlik yapılandırma hataları	Zayıflık referansı:	CWE-219, CWE-815(OWASP Top 10 2010 #A6)		
Test yöntemi: Web uygulama açıklık tarayıcısı, tasarım gözden geçirme				

Kural no: DK-11				
Kural tanımı: Web veya uygulama sunucularının, kendi sınırları dışında bulunan kaynak ve sistemlere uzak bağlantı ve erişimi varsayılan olarak engellenmelidir.				
Referans: OWASP ASVS 3.0/3.1 16.7	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Başka bir alandaki kaynaklara kontrolsüz başvuru	Zayıflık referansı:	CWE-610		
Test yöntemi: Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme				

Kural no: DK-12				
Kural tanımı: Uygulama, güvenilmeyen kaynaklardan alınmış veriyi çalıştırılabilir kod olarak koşturmamalıdır.				
Referans: OWASP ASVS 3.0/3.1 16.8	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Kod enjeksiyonu, genel enjeksiyon zayıflıkları	Zayıflık referansı:	CWE-94, CWE-74, OWASP Top 10 2013 #A1		
Test yöntemi: Dinamik analiz, karıştırmaya (fuzz) testi, veri akış analizi, statik analiz				

Kural no: DK-13				
Kural tanımı: Desteklenmeyen, güvensiz veya teknolojisi zaman aşımına uğramış istemci teknolojileri kullanılmamalıdır.				
Referans: OWASP ASVS 3.1 16.9	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Tehlikeli yazılımsal yöntem ve işlevlerin kullanıma açık olması	Zayıflık referansı:	CWE-749		
Test yöntemi: Kod kalitesi ölçüm aracı, kaynak kod gözden geçirme, tasarım gözden geçirme				

EK 2-Lahika-04: Oturum Yönetimi

Kural no: OY-01			
Kural tanımı: Kullanıcı oturumu kapattığında tüm oturumlar geçersiz hale getirilebilmelidir.			
Referans: OWASP ASVS 3.0.1 3.2	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Güvensiz oturum yönetimi	Zayıflık referansı: CWE-965		3
Test yöntemi: İşlevsel test, sızma testi			

Kural no: OY-02			
Kural tanımı: Oturumlar belirli bir süre etkinlik olmadığında kendiliğinden sonlanmalıdır.			
Referans: OWASP ASVS 3.0.1 3.3	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Oturumu süresi dolduğunda sonlandırma hatası	Zayıflık referansı: CWE-613		3
Test yöntemi: İşlevsel test, sızma testi			

Kural no: OY-03			
Kural tanımı: Kimlik doğrulamayla erişilen tüm sayfalardan oturum kapatma işlevine erişilebilmelidir.			
Referans: OWASP ASVS 3.0.1 3.5	Güvenlik Seviyesi:		2
Karşı koyduđu zayıflıklar: Veri ögesinin hatalı oturuma maruz kalması	Zayıflık referansı: CWE-488		3
Test yöntemi: İşlevsel test, sızma testi			

Kural no: OY-04			
Kural tanımı: Oturum kimliğinin URL, hata mesajları ve iz kayıtları içerisinde yer almaması sağlanmalıdır. URL içerisinde oturum kimliğinin yeniden yazılması engellenmelidir.			
Referans: OWASP ASVS 3.0.1 3.6	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Güvensiz çerez özellikleri, Açık Oturum Değişkenleri	Zayıflık referansı: CWE-565		3
Test yöntemi: Sızma testi, web uygulama açıklık tarayıcısı			

Kural no: OY-05			
Kural tanımı: Tüm kimlik doğrulama ve yeniden kimlik doğrulama işlemleri sonucunda yeni bir oturum ve yeni bir oturum kimliği üretilmelidir.			
Referans: OWASP ASVS 3.0.1 3.7	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Oturum sabitleme	Zayıflık referansı: CWE-384		3

Test yöntemi:

Sızma testi, web uygulama açıklık tarayıcısı

Kural no: OY-06**Kural tanımı:**

Yalnızca uygulama tarafından üretilen oturum kimliklerinin uygulamada aktif oturum kimliđi olarak kullanıldıđı doğrulanmalıdır.

Referans:

OWASP ASVS 3.0.1 3.10

Güvenlik Seviyesi:

1

2

3

Karşı koyduđu zayıflıklar:

Kaynak Doğrulama Hatası

Zayıflık referansı:

CWE-346

Test yöntemi:

Sızma testi, web uygulama açıklık tarayıcısı

Kural no: OY-07**Kural tanımı:**

Oturum kimlikleri yeterince uzun olmalı, rastgele olmalı ve etkin oturumlar içerisinde tekil olmalıdır.

Referans:

OWASP ASVS 3.0.1 3.11

Güvenlik Seviyesi:

1

2

3

Karşı koyduđu zayıflıklar:

Gözlemlenebilir durumun öngörülebilir olması, Yetersiz rasgele deđerlerin kullanımı

Zayıflık referansı:

CWE-341, CWE-330

Test yöntemi:

Vekil tabanlı ađ test yazılımı, web uygulama açıklık tarayıcısı

Kural no: OY-08**Kural tanımı:**

Uygulama tarafından etkin ve aynı zamanlı oturumların sayısı sınırlandırılabilir.

Referans:

OWASP ASVS 3.0.1 3.16

Güvenlik Seviyesi:

2

3

Karşı koyduđu zayıflıklar:

Hizmet engelleme

Zayıflık referansı:

OWASP Top Ten 2004 Category A9

Test yöntemi:

Sızma testi

Kural no: OY-09**Kural tanımı:**

Her bir kullanıcının uygulamadaki etkin oturumları görüntülenebilmeli, kullanıcı herhangi bir etkin oturumunu sonlandırabilmelidir.

Referans:

OWASP ASVS 3.0.1 3.17

Güvenlik Seviyesi:

2

3

Karşı koyduđu zayıflıklar:

Yetersiz oturum yönetimi

Zayıflık referansı:

OWASP Top Ten 2013 Category A2

Test yöntemi:

İşlevsel test

Kural no: OY-10**Kural tanımı:**

Parola deđiŐimi işleminde kullanıcıya tüm etkin oturumları sonlandırma seçeneđi sunulmalıdır.

Referans: OWASP ASVS 3.0.1 3.18	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: Oturumu süresi dolduğunda sonlandırma hatası	Zayıflık referansı:	CWE-613	
Test yöntemi: İşlevsel test			

Kural no: OY-11			
Kural tanımı: Oturum sonlandığında oturum ile ilgili tüm geçici depolama alanları ve çerezler uygulama tarafından silinmelidir.			
Referans: Appsec STIG SV-83863r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Süresi dolmuş veya bırakılmış kaynak üzerindeki işlem	Zayıflık referansı:	CWE-672	
Test yöntemi: Sekizli kod/ikili kod zayıflık analiz yazılımı, kaynak kodu zafiyet tarayıcısı			

Kural no: OY-12			
Kural tanımı: Web uygulamalarında oturum çerezlerinde HTTPOnly bayrağı etkin olmalıdır.			
Referans: Appsec STIG SV-84823r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: 'HttpOnly' Bayrağı Olmayan Hassas Çerez	Zayıflık referansı:	CWE-1004	
Test yöntemi: Web uygulama açıklık tarayıcısı, vekil tabanlı ağ test yazılımı			

Kural no: OY-13			
Kural tanımı: Uygulama her ürettiği oturum kimliğini yalnızca bir kez kullanmalıdır.			
Referans: Appsec STIG SV-84837r1_rule	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: Yeniden oynatma saldırıları, oturum sabitleme	Zayıflık referansı:	CWE-294, CWE-384	
Test yöntemi: Web uygulama açıklık tarayıcısı, vekil tabanlı ağ test yazılımı			

Kural no: OY-14			
Kural tanımı: Tanımlama bilgilerinde depolanan oturum kimliklerinin yolları, uygulama için uygun kısıtlayıcı bir değere ayarlanmalı ve kimlik doğrulama oturumu belirteçleri ayrıca "HttpOnly" ve "secure" olarak yapılandırılmalıdır.			
Referans: OWASP ASVS 3.0 3.12	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: HttpOnly bayrağı olmayan hassas çerez kullanımı	Zayıflık referansı:	CWE-1004	
Test yöntemi: Kaynak kod gözden geçirme, işlevsel test			

EK 2-Lahika-05: EriŐim Denetimi

Kural no: ED-01			
Kural tanımı: Kullanıcı sadece yetkilendirildiđi uygulama bileŐenlerine ve kaynaklara eriŐebilmeli ve bunları kullanabilmelidir.			
Referans: OWASP ASVS 3.0.1 4.1	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Yazılım, güvenlik açısından kritik bir kaynak için istenmeyen aktörler tarafından okunması veya deđiŐtirilmesi için izinler belirtir.	Zayıflık referansı:	CWE-269	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-02			
Kural tanımı: Uygulama bir kullanıcının diđer kullanıcılara ait hassas bilgilere eriŐimini engellemelidir.			
Referans: OWASP ASVS 3.0.1 4.4	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Yetersiz eriŐim denetimi	Zayıflık referansı:	CWE-284	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-03			
Kural tanımı: Uygulama dizinlerde dolaŐıma gerekli haller dışında izin vermemelidir. Var olan dosyalara veya dosyalar hakkındaki üst verilere eriŐilememedir.			
Referans: OWASP ASVS 3.0.1 4.5	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Dizin dolaŐımı, bađıl izin yoluna dolaŐım, mutlak izin adına dolaŐım	Zayıflık referansı:	CWE-21, CWE-23, CWE-36	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-05			
Kural tanımı: Uygulama, fonksiyonlara, kaynaklara ya da verilere eriŐim sürelerini, kullanım oranlarını veya kullanım sıklıđını deđiŐtirebilmelidir.			
Referans: OWASP ASVS 3.0.1 4.14	Güvenlik Seviyesi:		2
KarŐı koyduđu zayıflıklar: Kaynakların sınır konulmadan veya yavaŐlatma olmadan ayrılması	Zayıflık referansı:	CWE-770	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-07			
Kural tanımı: Uygulama içerik duyarlı erişim denetimi yapabilmelidir (zaman, konum, öznitelikler vb.).			
Referans: OWASP ASVS 3.0.1 4.16	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Yetersiz yetkilendirme	Zayıflık referansı:	CWE-285	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-08			
Kural tanımı: Uygulama verilere ve kaynaklara erişim için, erişim denetimi politikalarına uygun şekilde yetkilendirmeyi sağlamalıdır.			
Referans: Appsec STIG SV-83951r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Yetersiz yetkilendirme	Zayıflık referansı:	CWE-285	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-09			
Kural tanımı: Uygulama veri akışı kontrol politikalarına dayalı olarak sistem içi veri akışı için erişim denetimi yapmalıdır.			
Referans: Appsec STIG SV-83955r1_rule	Güvenlik Seviyesi:		3
Karşı koyduğu zayıflıklar: Bilgi sızması/açığa çıkması	Zayıflık referansı:	CWE-200	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-10			
Kural tanımı: Uygulama veri akışı kontrol politikalarına dayalı olarak sistemler arası veri akışı için erişim denetimi yapılmalıdır.			
Referans: Appsec STIG SV-83957r1_rule	Güvenlik Seviyesi:		3
Karşı koyduğu zayıflıklar: Bilgi sızması/açığa çıkması	Zayıflık referansı:	CWE-200	
Test yöntemi: Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler			

Kural no: ED-11			
Kural tanımı: Uygulama yetkisiz kullanıcıların, güvenlikle ilişkili işlevleri kullanmasına, güvenlik önlemlerini değiştirmesine, kapatmasına, atlatmasına izin vermemelidir.			
Referans: Appsec STIG SV-83959r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Yetki bağlamı değiştirme hatası	Zayıflık referansı:	CWE-270	

Test yöntemi:

Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler

Kural no: ED-12**Kural tanımı:**

Uygulama denetim kayıtlarının yetkisiz okunmasını, deęiŐtirilmesini ve silinmesini engellemelidir.

Referans:

Appsec STIG SV-84105r1_rule

Güvenlik Seviyesi:

1

2

3

KarŐı koyduęu zayıflıklar:

Kritik kaynaklar için hatalı izin ataması

Zayıflık referansı:

CWE-732

Test yöntemi:

Kaynak kodu gözden geçirme, Kayıtları gözden geçirme

Kural no: ED-13**Kural tanımı:**

Uygulama, kurumsal bilgi sistemlerinde saklanan ve kendi sorumluluęunda olmayan verilerin deęiŐtirilebilmesini engellemelidir.

Referans:

Appsec STIG SV-84849r1

Güvenlik Seviyesi:

2

3

KarŐı koyduęu zayıflıklar:

Sistem verisinin yetkisiz bir kontrol alanından erişimine izin verme

Zayıflık referansı:

CWE-497

Test yöntemi:

Tasarım gözden geçirme, Kaynak kodu gözden geçirme, test kapsama analizi, biçimsel yöntemler

EK 2-Lahika-06: Güvenli Kurulum ve Yapılandırma

Kural no: GK-01			
Kural tanımı: Uygulama, güvenlik güncellemeleri ve yamaları yapılmıő bileőenlerden oluőmalıdır.			
Referans: OWASP ASVS 3.0.1 19.1	Güvenlik Seviyesi:	1	2
Karőı koyduėu zayıflıklar: Yapılandırma hataları ve eksiklikleri	Zayıflık referansı: OWASP Top Ten 2013 Category A5		3
Test yöntemi: Sızma testi, gözden geçirme			

Kural no: GK-02			
Kural tanımı: Uygulama sunucuları ve veritabanı sunucuları gibi bileőenlerin arasındaki iletiőim Őifreli olmalıdır.			
Referans: OWASP ASVS 3.0.1 19.2	Güvenlik Seviyesi:		2
Karőı koyduėu zayıflıklar: Yazılım, hassas veya güvenlik aısından kritik olan verileri yetkisiz kiőilerce dinlenebilen bir iletiőim kanalında düz metin halinde iletir.	Zayıflık referansı: CWE-319		3
Test yöntemi: Gözden geçirme, aė izleyici/protokol test yazılımı, iőlevsel test			

Kural no: GK-03			
Kural tanımı: Uygulama sunucuları ve veritabanı sunucuları gibi bileőenlerin arasındaki iletiőimde ihtiya duyulan en az yetki sahip hesaplar kullanılmalıdır.			
Referans: OWASP ASVS 3.0.1 19.3	Güvenlik Seviyesi:		2
Karőı koyduėu zayıflıklar: Gereksiz yetkilerle alıőtırma, En az yetki prensibine uyulmaması	Zayıflık referansı: CWE-250, CWE 272		3
Test yöntemi: Web uygulama aıklık tarayıcısı, uygulama aıklık tarayıcısı, tasarım gözden geçirme, iőlevsel test, negatif test, kaynak kod analizi, kaynak kod gözden geçirme			

Kural no: GK-04			
Kural tanımı: Uygulama kurulumları, yeterince korunaklı, korumalı ve ayrıőtırılmıő Őekilde yapılmalıdır.			
Referans: OWASP ASVS 3.0.1 19.4	Güvenlik Seviyesi:		2
Karőı koyduėu zayıflıklar: Yetersiz ayrıőtırma	Zayıflık referansı: CWE 265		3
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, negatif test, kaynak kod analizi			

Kural no: GK-08				
Kural tanımı: Sistem seviyesinde erişimi olan diller ile geliştirilmiş uygulamalar ASLR, DEP ve güvenlik denetimleri gibi tüm güvenlik bayrakları etkin olacak şekilde derlenmiş olmalıdır.				
Referans: OWASP ASVS 3.0.1 19.9	Güvenlik Seviyesi:		2	3
Karşı koyduğu zayıflıklar: Bellek Arabelleđi Sınırları İçindeki İşlemlerin Uygunsuz Kısıtlanması	Zayıflık referansı: CWE-119			
Test yöntemi: Kaynak kod zayıflık analiz yazılımı, sekizli kod/ikili kod zayıflık analiz yazılımı, kaynak kod gözden geçirme, negatif test				

Kural no: GK-09				
Kural tanımı: Uygulama, yapılandırma deđişiklikleri ile ilgili erişimleri kısıtlamalı ve yapılandırma deđişiklikleri için iz kayıtları oluşturmalıdır.				
Referans: Appsec STIG SV-84127r1_rule	Güvenlik Seviyesi:		2	3
Karşı koyduğu zayıflıklar: Kritik kaynaklar için hatalı izin ataması, Geçersiz/hatalı yetkilendirme	Zayıflık referansı: CWE-732			
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme, negatif test, kaynak kod analizi				

Kural no: GK-11				
Kural tanımı: Uygulamada kurulum ile gelen gereksiz hesaplar olmamalıdır.				
Referans: Appsec STIG SV-85023r1_rule	Güvenlik Seviyesi:		2	3
Karşı koyduğu zayıflıklar: Gereksiz Ayrıcalıklarla Yürütme	Zayıflık referansı: CWE-250			
Test yöntemi: Gözden geçirme, negatif test				

Kural no: GK-14				
Kural tanımı: Uygulama, sistem kaynakların azalması durumunda bir yöneticiye uyarı vermelidir.				
Referans: Appsec STIG SV-85037r1_rule	Güvenlik Seviyesi:		2	3
Karşı koyduğu zayıflıklar: Kontrolsüz Kaynak Tüketimi ("Kaynak Tüketimi"), Hizmet Reddi	Zayıflık referansı: CWE-400, OWASP Top Ten 2004 Category A9			
Test yöntemi: İşlevsel test, negatif test, kaynak kod analizi, kaynak kod gözden geçirme				

Kural no: GK-15			
Kural tanımı: Uygulama, güncelleme bildirimlerini ya da güvenlik uyarılarını e-posta, SMS veya alternatif iletişim kanallarıyla iletebilmelidir.			
Referans: Appsec STIG SV-85039r1_rule	Güvenlik Seviyesi:		3
Karşı koyduđu zayıflıklar: Hizmet Reddi	Zayıflık referansı: OWASP Top Ten 2004 Category A9		
Test yöntemi: Gözden geçirme			

Kural no: GK-16			
Kural tanımı: Alternatif ve güvenli olmayan erişim yolları ile uygulamaya erişilememelidir.			
Referans: OWASP ASVS 3.0 18.1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Korunmayan alternatif kanal	Zayıflık referansı: CWE-420		
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme			

Kural no: GK-17			
Kural tanımı: Uygulama, başarısız sistem başlatma, başarısız sonlandırma veya başarısız kapatma gibi işlemlerde güvenli bir duruma geçmelidir.			
Referans: Appsec STIG SV-84843r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Güvensiz Olarak Başarısız Olma ('Başarısız durumda açık duruma düşme')	Zayıflık referansı: CWE-636		
Test yöntemi: Negatif test, işlevsel test, gözden geçirme			

EK 2-Lahika-07: Güçlü Kriptografik Mekanizmaların Kullanımı

Kural no: KM-01			
Kural tanımı: Tüm kriptografik modüllerin, güvenli bir şekilde hataya düŐtüğü doğrulanmalıdır. Hata yönetimi "Oracle Padding" atağına imkan tanımayacak şekilde olmalıdır.			
Referans: OWASP ASVS 3.0.1 7.2	Güvenlik Seviyesi:	1	2
KarŐı koyduğı zayıflıklar: Güvensiz Olarak Başarısız Olma ('Başarısız durumda açık duruma düşme')	Zayıflık referansı: CWE-636		3
Test yöntemi: Negatif test, işlevsel test, gözden geçirme			

Kural no: KM-02			
Kural tanımı: Tüm rastgele üretilen sayılar, dosya isimleri, global eşsiz deęerler (GUID) ve karakter dizilerinin saldırgan için tahmin edilemez olması sağlanmalıdır. Rastgele sayıların yüksek entropiye sahip olarak üretilmelidir			
Referans: OWASP ASVS 3.0.1 7.6	Güvenlik Seviyesi:		2
KarŐı koyduğı zayıflıklar: Kriptografik hatalar	Zayıflık referansı: CWE-310		3
Test yöntemi: Negatif test, işlevsel test, gözden geçirme			

Kural no: KM-06			
Kural tanımı: Kriptografik servisleri kullanan bileŐenlerin anahtarlara erişimi olmamalıdır. Kriptografik prosesler ve ana anahtarlar izole edilmelidir.			
Referans: OWASP ASVS 3.0.1 7.11	Güvenlik Seviyesi:		3
KarŐı koyduğı zayıflıklar: Kriptografik hatalar, Anahtar yönetimi hataları	Zayıflık referansı: CWE-310, CWE-320		
Test yöntemi: Negatif test, işlevsel test, biçimsel doğrulama			

Kural no: KM-07			
Kural tanımı: Tüm anahtar ve şifreler kullanımları tamamlandıđında, tamamen sıfırlanarak yok edilmelidir.			
Referans: OWASP ASVS 3.0.1 7.13	Güvenlik Seviyesi:		2
KarŐı koyduğı zayıflıklar: Kriptografik hatalar	Zayıflık referansı: CWE-310		3
Test yöntemi: Negatif test, işlevsel test, biçimsel doğrulama			

Kural no: KM-08			
Kural tanımı: Tüm anahtar ve parolalar deęiŐtirilebilir olmalıdır ve kurulum esnasında oluşturulmalı veya deęiŐtirilmelidir.			
Referans: OWASP ASVS 3.0.1 7.14	Güvenlik Seviyesi:		2
			3

Karşı koyduđu zayıflıklar: Kriptografik hatalar	Zayıflık referansı: CWE-310
Test yöntemi: Negatif test, işlevsel test, biçimsel doğrulama	

Kural no: KM-13				
Kural tanımı: Uygulamada şifreleme, anahtar deđiŐimi, dijital imzalama veya özet alma gibi fonksiyonlar bulunuyorsa TS ISO/IEC 19790-24759 onaylı kriptografik modüller ve rasgele sayı üreticileri kullanılmalıdır.				
Referans: Appsec STIG SV-84839r1_rule	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Kriptografik hatalar, Kırılmış veya riskli kriptografik algoritmaların kullanımı	Zayıflık referansı: CWE-310, CWE-327			
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme				

Kural no: KM-14				
Kural tanımı: Uygulama anahtar deđiŐimi yapacaksa, anahtar deđiŐimi için güvenli kanal oluşturulmadan önce uç noktalar için kimlik denetimi yapılmalıdır ve şifreleme yapılmalıdır.				
Referans: Appsec STIG SV-84983r1_rule	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Kriptografik hatalar, Anahtar yönetimi hataları	Zayıflık referansı: CWE-310, CWE-320			
Test yöntemi: Negatif test, işlevsel test, gözden geçirme				

Kural no: KM-15				
Kural tanımı: Uygulama saklanan kurumsal verilerin yetkisiz bir şekilde deđiŐtirilmesini engellemek için kriptografik yöntemler uygulamalıdır.				
Referans: Appsec STIG SV-84849r1_rule	Güvenlik Seviyesi:		2	3
Karşı koyduđu zayıflıklar: Kriptografik hatalar, Gerekli kriptografik adımın atlanması	Zayıflık referansı: CWE-310, CWE-325			
Test yöntemi: Negatif test, işlevsel test, gözden geçirme				

EK 2-Lahika-08: Veri Koruma

Kural no: VK-01			
Kural tanımı: Tüm formlarda istemci tarafında yapılan ön bellekleme işlevselliği önemli veriler için kapatılmalıdır.			
Referans: OWASP ASVS 3.0.1 9.1	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Yetkisiz bilgilere erişim, servis dışı bırakma	Zayıflık referansı:	CWE-524, CWE-525	
Test yöntemi: Vekil tabanlı ağ test yazılımı, web uygulama açıklık tarayıcısı			

Kural no: VK-02			
Kural tanımı: Uygulama-istemci arasındaki hassas veri iletişim için HTTP başlığı veya gövdesi kullanılmalıdır.			
Referans: OWASP ASVS 3.0.1 9.3	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Siteler arası komut dosyası çalıştırma (XSS)	Zayıflık referansı:	CWE-79	
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, web uygulama açıklık tarayıcısı, vekil tabanlı ağ test yazılımı			

Kural no: VK-03			
Kural tanımı: Sunucu üzerinde saklanan önemli verilerin ön belleklenmiş ya da geçici üretilmiş kopyaları şifreli ve güvenli bir şekilde saklanmalıdır.			
Referans: OWASP ASVS 3.0.1 9.5	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: saldırı gerçekleştirmesine izin veren verinin ortaya çıkması	Zayıflık referansı:	CWE-320	
Test yöntemi: Sekizli kod/ikili kod zayıflık analiz yazılımı, tasarım gözden geçirme, gözden geçirme			

Kural no: VK-04			
Kural tanımı: Uygulama, saklama gereksinimi sona erdikten sonra önemli verileri güvenlik sonunu yaratmayacak şekilde silinmelidir.			
Referans: OWASP ASVS 3.0.1 9.6	Güvenlik Seviyesi:		3
Karşı koyduğu zayıflıklar: saldırı gerçekleştirmesine izin veren verinin ortaya çıkması	Zayıflık referansı:	CWE-212	
Test yöntemi: Sekizli kod/ikili kod zayıflık analiz yazılımı, kaynak kod gözden geçirme, tasarım gözden geçirme, gözden geçirme			

Kural no: VK-05				
Kural tanımı: Sunucuya gelen isteklerin öngörülmeleyen bir sayıda ya da büyüklükte olup olmadığı kontrol edilebilmelidir.				
Referans: OWASP ASVS 3.0.1 9.8	Güvenlik Seviyesi:			3
Karşı koyduđu zayıflıklar: Kontrolsüz Kaynak Tüketimi ('Kaynak Tüketimi'), Yetersiz kaynak havuzu	Zayıflık referansı: CWE-400, CWE-410			
Test yöntemi: Negatif test				

Kural no: VK-06				
Kural tanımı: İz kaydı bilgilerinin erişimi sadece yetkili kullanıcılara açık tutulmalı ve güvenlik denetimi yapılmalıdır.				
Referans: OWASP ASVS 3.0.1 9.10	Güvenlik Seviyesi:		2	3
Karşı koyduđu zayıflıklar: Hassas kullanıcı verilerini iz kaydına ekleme, genellikle saldırganlara bilgi edinmek için daha az korumalı bir yol sağlar.	Zayıflık referansı: CWE-532			
Test yöntemi: Kaynak kod gözden geçirme, tasarım gözden geçirme, gözden geçirme				

Kural no: VK-07				
Kural tanımı: Bellekte tutulan önemli veriler gereksinimi sona erdiğinde güvenlik ihlali oluşturamayacak şekilde silinmelidir.				
Referans: OWASP ASVS 3.0.1 9.11	Güvenlik Seviyesi:		2	3
Karşı koyduđu zayıflıklar: Kullanımdan sonra hafıza silinmezse, hafıza yeniden tahsis edildiğinde istenmeyen aktörlerin verileri okumasına izin verebilir.	Zayıflık referansı: CWE-226			
Test yöntemi: Sekizli kod/ikili kod zayıflık analiz yazılımı, kaynak kod gözden geçirme, tasarım gözden geçirme, gözden geçirme				

EK 2-Lahika-09: Hata Ele Alma ve Kayıt

Kural no: HK-01			
Kural tanımı: Uygulama, hassas veri ve kişisel verileri içeren hata mesajı veya iz kaydı üretmemelidir.			
Referans: OWASP ASVS 3.0 8.1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Yazılım, ortamı, kullanıcıları veya ilişkili verileri hakkında önemli bilgiler içeren bir hata mesajı üretir.	Zayıflık referansı: CWE-209, CWE-532, CWE-544		3
Test yöntemi: Gözden geçirme, Kaynak kodu gözden geçirme			

Kural no: HK-02			
Kural tanımı: Uygulamada hata durumunda erişim varsayılan olarak engellenmelidir.			
Referans: OWASP ASVS 3.0.1 8.2	Güvenlik Seviyesi:		2
Karşı koyduđu zayıflıklar: Günlük dosyalarına yazılan bilgiler hassas bir nitelikte olabilir ve bir saldırganı değerli yol gösterebilir veya hassas kullanıcı bilgilerini ortaya çıkarabilir.	Zayıflık referansı: CWE-276		3
Test yöntemi: Sızma testi, negatif test			

Kural no: HK-03			
Kural tanımı: Uygulama, tanımlanan güvenlik olayları listesinde tanımlanan tüm olayların başarılı ve başarısızlık durumları için iz kayıtları oluşturabilmelidir.			
Referans: OWASP ASVS 3.0.1 8.3	Güvenlik Seviyesi:		2
Karşı koyduđu zayıflıklar: Zayıflıklara ve saldırılara neden olabilecek durumların tespit edilememesi	Zayıflık referansı:		3
Test yöntemi: Gözden geçirme			

Kural no: HK-04			
Kural tanımı: İz kayıtlarında olayların zaman sıralamasına ilişkin araştırma yapılabilecek şekilde zaman bilgisi yer almalıdır.			
Referans: OWASP ASVS 3.0.1 8.4	Güvenlik Seviyesi:		2
Karşı koyduđu zayıflıklar: Yetersiz iz kaydı	Zayıflık referansı: CWE-778		3
Test yöntemi: Gözden geçirme			

Kural no: HK-05			
Kural tanımı: İz kayıtlarını okuyan araçlarda istenilmeyen bir işlemi yapacak kayıt üretilmemelidir.			
Referans: OWASP ASVS 3.0.1 8.5	Güvenlik Seviyesi:		3
Karşı koyduđu zayıflıklar: İz kayıtlarında yetersiz çıktı kontrolü	Zayıflık referansı: CWE-117		
Test yöntemi: Gözden geçirme			

Kural no: HK-06			
Kural tanımı: Uygulama tarafından üretilen iz kayıtları hassas bilgi içermemelidir.			
Referans: OWASP ASVS 3.0.1 8.7	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: Hassas kullanıcı verilerini iz kaydına ekleme,	Zayıflık referansı: CWE-532		
Test yöntemi: Gözden geçirme			

Kural no: HK-07			
Kural tanımı: İz kaydı bilgileri 5651 sayılı kanuna uygun şekilde elektronik olarak imzalanmalıdır.			
Referans: OWASP ASVS 3.0.1 8.9	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: İz kayıtlarında yetersiz çıktı kontrolü	Zayıflık referansı: CWE-117		
Test yöntemi: Gözden geçirme			

Kural no: HK-08			
Kural tanımı: Uygulama, uygulama sunucusu ele geçirildiğinde iz kayıtlarının deđiŐtirilmesini veya ortadan kaldırılmasını izin vermemelidir.			
Referans: OWASP ASVS 3.0.1 8.12	Güvenlik Seviyesi:		3
Karşı koyduđu zayıflıklar: İz kayıtlarında yetersiz çıktı kontrolü	Zayıflık referansı: CWE-117		
Test yöntemi: Gözden geçirme			

EK 2-Lahika-10: İletişim Güvenliđi

Kural no: IG-01			
Kural tanımı: Güvenilen bir sertifika otoritesinden her Transport Layer Security (TLS) sunucu sertifikasına bir güven zinciri oluşturulabilmeli ve her sunucu sertifikası geçerli olmalıdır.			
Referans: OWASP ASVS 3.0.1 10.1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Sertifikanın güven zincirini hatalı doğrulama	Zayıflık referansı:	CWE-296	
Test yöntemi: İşlevsel test			

Kural no: IG-02			
Kural tanımı: Kimlik doğrulaması yapılmıő, hassas veriler ya da işlevler içeren ve güvensiz ya da şifrelenmemiő protokollerle yapılan tüm bağlantılar (iç ve dış) için TLS protokolünün yaygın kullanılan son sürümü üzerinden yapılmalıdır.			
Referans: OWASP ASVS 3.0.2 10.3	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Hassas verinin açık olarak iletilmesi	Zayıflık referansı:	CWE-319, CWE 311(CWE/ SANS Top 25 2011 #8)	
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, kaynak kodu gözden geçirme, ağ trafiđi izleme			

Kural no: IG-03			
Kural tanımı: Uygulama iç iletişimindeki TLS bağlantı hatası durumları için iz kaydı oluşturulmalıdır.			
Referans: OWASP ASVS 3.0.3 10.4	Güvenlik Seviyesi:		3
Karşı koyduđu zayıflıklar: Güvenlikle ilgili bilginin kaydedilmemesi	Zayıflık referansı:	CWE-778	
Test yöntemi: İz kaydı gözden geçirme, tasarım gözden geçirme, işlevsel test			

Kural no: IG-04			
Kural tanımı: Hassas bilgileri ve işlevleri kapsayan dış sistemlerle olan tüm bağlantılar, kimlik doğrulamasına tabi olmalıdır.			
Referans: OWASP ASVS 3.0.5 10.6	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: Kritik bilgi ve işlevlere kimlik doğrulama olmadan erişim	Zayıflık referansı:	CWE-306	
Test yöntemi: Kaynak kodu zayıflık tarayıcısı, kaynak kodu gözden geçirme, negatif test, test kapsama analizi			

Kural no: IG-05			
Kural tanımı: Uygulamanın TLS gereklemesi, onaylanmış ve en güvenli alıŐma modunda alıŐacak Őekilde yapılandırılmış olmalıdır.			
Referans: OWASP ASVS 3.0.6 10.8	Güvenlik Seviyesi:		3
KarŐı koyduđu zayıflıklar: İletişimde güvensiz algoritma seçimi	Zayıflık referansı: CWE-757		
Test yöntemi: Web uygulama açıklık tarayıcısı, tasarım gözden geçirme			

Kural no: IG-06			
Kural tanımı: HTTP açık anahtar sabitlemesi ürün ve yedekleme açık anahtarlarıyla gerekleştirilmelidir (Örnek https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning)			
Referans: OWASP ASVS 3.0.7 10.10	Güvenlik Seviyesi:		3
KarŐı koyduđu zayıflıklar: Kimlik doğrulamasız anahtar deđiŐimi	Zayıflık referansı: CWE-322		
Test yöntemi: Vekil tabanlı ađ test yazılımı, ađ izleyici yazılım, işlevsel test			

Kural no: IG-07				
Kural tanımı: HTTP Sert TaŐıma Güvenliđi (HTTP Strict Transport Security, HSTS) üst bilgileri tüm isteklerde ve alt alanlarda yer almalıdır.				
Referans: OWASP ASVS 3.0.8 10.11	Güvenlik Seviyesi:	1	2	3
KarŐı koyduđu zayıflıklar: Güvenli öznitelik olmadan HTTPS içerisinde çerez kullanımı, iletişimde güvensiz algoritma seçimi	Zayıflık referansı: CWE-614, CWE 757			
Test yöntemi: Vekil tabanlı ađ test yazılımı, ađ izleyici yazılım, işlevsel test				

Kural no: IG-08			
Kural tanımı: Operasyonel web sitesi URL bilgisi, web tarayıcı sağlayıcıları tarafından tutulan önceden yüklenmiş TaŐımalı Güvenlik Alanları listelerinde yer almalıdır (Örneđin: https://www.chromium.org/hsts)			
Referans: OWASP ASVS 3.0.9 10.12	Güvenlik Seviyesi:		3
KarŐı koyduđu zayıflıklar: Oltalama saldırıları	Zayıflık referansı: CWE-601		
Test yöntemi: Vekil tabanlı ađ test yazılımı, ađ izleyici yazılım, işlevsel test			

Kural no: IG-09			
Kural tanımı: Uygulamada, ađı dinleyen saldırganların trafiđi kaydetmesini engellemek için ileri gizlilik Őifrelemeleri kullanılmalıdır.			

Referans: OWASP ASVS 3.0.10 10.13	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Yetersiz güçte şifreleme	Zayıflık referansı:	CWE-326		
Test yöntemi: Tasarım gözden geçirme, kaynak kod gözden geçirme				

Kural no: IG-10				
Kural tanımı: Uygulama, Çevrimiçi Sertifika Durum Protokolü Damgalama (OCSP stapling) gibi yöntemlerle sertifika iptal denetimi gerçekleştirebilecek şekilde yapılandırılmalıdır (https://en.wikipedia.org/wiki/OCSP_stapling)				
Referans: OWASP ASVS 3.0.11 10.14	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Sertifika iptal denetimini yetersiz yapma	Zayıflık referansı:	CWE-299		
Test yöntemi: Vekil tabanlı ağ test yazılımı, ağ izleyici yazılım, işlevsel test				

Kural no: IG-11				
Kural tanımı: Sertifikalarda ve sertifikanın tüm hiyerarşisinde yalnızca güçlü algoritmalar ve protokoller kullanılmalıdır.				
Referans: OWASP ASVS 3.0.12 10.15	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Zayıf ve standart olmayan kriptografik mekanizmalar	Zayıflık referansı:	CWE-327		
Test yöntemi: Güvenlik kütüphaneleri doğrulama, ağ servis tarama ve tanılama, işlevsel test, gözden geçirme				

Kural no: IG-12				
Kural tanımı: Özellikle yaygın yapılandırmalar, şifreler ve algoritmalar güvensiz hale geldiğinden, TLS ayarları en güncel yapılandırma önerileri ile uyumlu olmalıdır (https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet).				
Referans: OWASP ASVS 3.0.13 10.16	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Etkileşimli protokollerde her iki tarafın desteklediği en güçlü şifreleme ve kimlik doğrulama algoritmalarının seçilmemesi	Zayıflık referansı:	CWE-757		
Test yöntemi: Güvenlik kütüphaneleri doğrulama, ağ servis tarama ve tanılama, işlevsel test, gözden geçirme				

Kural no: IG-13				
Kural tanımı: Kullanıcı erişimi için kimlik doğrulaması gerektiren uygulamalar, kullanıcı tarafından başlatılan iletişim oturumunu sonlandırmak için bir oturum kapatma yeteneği sağlamalıdır ve bu yetenek oturumun sonunda iletişim oturumu ile ilişkili tüm ağ bağlantılarını sona erdirmelidir.				
Referans: Appsec STIG	Güvenlik Seviyesi:		2	3

Karşı koyduđu zayıflıklar: Eski oturum kimlik bilgilerinin yetkilendirme için kullanılması	Zayıflık referansı: CWE-613
Test yöntemi: Vekil tabanlı ağ test yazılımı, işlevsel test	

Kural no: IG-15			
Kural tanımı: Uygulama, kimliđi doğrulanmış iletişim oturumlarının güvenilir olarak sonlandırıldığını belirten ve kolay anlaşılabilen bir çıkış iletisi görüntülemelidir.			
Referans: Appsec STIG	Güvenlik Seviyesi:	2	3
Karşı koyduđu zayıflıklar: Oturum kapatmayla ilgili genel zayıflıklar	Zayıflık referansı: CWE-678		
Test yöntemi: Web uygulama açıklık tarayıcısı, ağ izleyici/protokol test yazılımı, tasarım gözden geçirme, işlevsel test, negatif test			

EK 2-Lahika-11: İŐ Mantiđı

Kural no: IM-01				
Kural tanımı: Uygulama iŐ mantıđını dođru bir Őekilde gerŐekleŐtirmeli, iŐ mantıđındaki akıŐlar yazılımda beklenen sırada gerŐekleŐmeli, gereken adımlar atlanmamalı, adımların insanların yapabileceđi süreler içinde gerŐekleŐtirildiđi kontrol edilmeli ve őkok yüksek sıklıkla gönderilen istekler tespit edilmelidir.				
Referans: OWASP ASVS 3.0.1 15.1	Güvenlik Seviyesi:		2	3
KarŐı koyduđu zayıflıklar: DavranıŐa bađlı iŐ akıŐının yetersiz olarak gerŐekleŐtirilmesi	Zayıflık referansı: CWE-841			
Test yöntemi: Tasarım Gözden Geçirme, kaynak kod gözden geçirme, iŐlevsel test				

Kural no: IM-02				
Kural tanımı: Uygulamada őkewitli sınırlamalar tanımlanabilmeli ve bu sınırlamalar her bir kullanıcı için uygulanabilecek Őekilde tanımlanabilmelidir. Uygulama, ihlal veya olađanüstü durumlar için dođru uyarı ve otomatik tepkiler verebilmelidir.				
Referans: OWASP ASVS 3.0.1 15.2	Güvenlik Seviyesi:		2	3
KarŐı koyduđu zayıflıklar: EtkileŐim frekansının yetersiz olarak denetlenmesi	Zayıflık referansı: CWE-799			
Test yöntemi: Negatif test, gözden geçirme				

EK 2-Lahika-12: Kötücül İşlemleri Engelleme

Kural no: KI-01			
Kural tanımı: Uygulama, harici bir program çalıştırıyorsa, organizasyon politikalarına doğrultusunda yetkisiz olarak program çalıştırılmasını ve yapılandırılmasını engellemelidir.			
Referans: Appsec STIG SV-84137r1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Gereksiz yetkilerle çalıştırma	Zayıflık referansı:	CWE-250	
Test yöntemi: Negatif test, işlevsel test, gözden geçirme			

Kural no: KI-04			
Kural tanımı: Uygulama hassas bilgileri formlarda bulunan gizli alanlarda saklamamalıdır.			
Referans: Appsec STIG SV-84877r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Kullanıcıyı giz bilginin açıklanması olarak saklanması	Zayıflık referansı:	CWE-317	
Test yöntemi: Kaynak kod gözden geçirme, vekil tabanlı ağ test yazılımı, ağ izleme yazılımı			

Kural no: KI-05			
Kural tanımı: Uygulama Cross-Site Request Forgery (CSRF)'dan kaynaklanan açıklıklardan korunma mekanizmasına sahip olmalıdır.			
Referans: Appsec STIG SV-84881r1_rule	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Siteler Arası Talep Sahteciliđi (CSRF)	Zayıflık referansı:	CWE-352	
Test yöntemi: Web uygulama açıklık tarayıcısı, fuzz testi, kaynak kod zayıflık analizi			

Kural no: KI-07			
Kural tanımı: Uygulama veri gösterim biçimlerindeki farklılıklardan kaynaklanan açıklıklardan korunmalıdır.			
Referans: Appsec STIG SV-84885r1_rule	Güvenlik Seviyesi:		2
Karşı koyduđu zayıflıklar: Veriyi, gösterim için hazırlamadan önde doğrulamama	Zayıflık referansı:	CWE-180	
Test yöntemi: Dinamik test, negatif test, web uygulama açıklık tarayıcısı			

EK 2-Lahika-13: Mobil Uygulama Güvenliđi

Kural no: MG-01			
Kural tanımı: Gizli anahtarlar, API token ya da Őifreler mobil uygulamalarda dinamik olarak oluŐturulmalıdır.			
Referans: OWASP ASVS 3.0 17.4	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Varlık Kimlik Doğrulaması Olmadan Anahtar DeđiŐimi	Zayıflık referansı: CWE-322	3	
Test yöntemi: Kaynak kod gözden geçirme			

Kural no: MG-02			
Kural tanımı: Uygulama gerekli iŐlev ve kaynaklar için minimum izne sahip olmalıdır.			
Referans: OWASP ASVS 3.0 17.6	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Gereksiz yetkilerle çalıŐtırma	Zayıflık referansı: CWE-250	3	
Test yöntemi: Kaynak kod gözden geçirme, negatif test			

Kural no: MG-03			
Kural tanımı: Uygulama, aynı cihazdaki diđer uygulamalar için hassas bilgileri dıŐarı aktarmamalıdır.			
Referans: OWASP ASVS 3.0 17.9	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Android Uygulaması BileŐenlerinin Hatalı DıŐa Aktarımı	Zayıflık referansı: CWE-926	3	
Test yöntemi: Kaynak kod gözden geçirme, negatif test			

Kural no: MG-04			
Kural tanımı: Uygulamanın içerik sađlayıcıları, aktiviteleri tüm girdileri doğrulamalıdır.			
Referans: OWASP ASVS 3.1 17.11	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Android Uygulaması BileŐenlerinin Hatalı DıŐa Aktarımı	Zayıflık referansı: CWE-926	3	
Test yöntemi: Negatif test			

EK 2-Lahika-14: Girdi ve Çıktı Süzme

Kural no: GC-02			
Kural tanımı: Uygulamanın çalışma ortamı, bellek taşması saldırılarına dayanıklı olmalıdır veya mevcut güvenlik mekanizmaları bellek taşmasını engellemelidir.			
Referans: OWASP ASVS 3.0 5.1	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Ayrılmıő bellek alanı sonrasına erişme, Klasik bellek taşması - girdinin boyutunu hesaplamadan bellek alanı kopyalama	Zayıflık referansı:	CWE-788 CWE-120	3
Test yöntemi: Sekizli kod/ikili kod analizi, statik analiz, web uygulama açıklık tarayıcısı, kaynak kodu zayıflık tarayıcısı, kullanılan dil ve kütüphaneleri gözden geçirme			

Kural no: GC-03			
Kural tanımı: Sunucuda yapılan girdi doğrulama hataları, isteđin reddi ile sonuçlanmalı ve iz kaydı oluşturulmalıdır.			
Referans: OWASP ASVS 3.0 5.3	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Hatalı girdi doğrulama	Zayıflık referansı:	CWE-20	3
Test yöntemi: İşlevsel test, tasarım gözden geçirme			

Kural no: GC-04			
Kural tanımı: Uygulama tarafından, istemci ve sunucu tarafında, kabul edilen her bir veri tipi için girdi doğrulama denetimi yapılmalıdır.			
Referans: OWASP ASVS 3.0 5.5	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Girdi doğrulama çerçevesi kullanmamak, hatalı girdi doğrulama	Zayıflık referansı:	CWE-554, CWE-20, OWASP Top Ten 2004 #A1	3
Test yöntemi: Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod analizi, kaynak kod gözden geçirme			

Kural no: GC-05			
Kural tanımı: Bütün veritabanı sorguları, parametre olarak yapılmalı ve veritabanına erişimde kullanılan dile karşı (SQL, NoSQL vb.) enjeksiyon saldırılarını önleyebilecek denetimler yapılmalıdır.			
Referans: OWASP ASVS 3.0 5.10	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: SQL saldırısında kullanılan özel karakterlerin engellenmemesi ('SQL enjeksiyonu')	Zayıflık referansı:	CWE-89	3
Test yöntemi: Veritabanı açıklık tarama aracı, web uygulama zayıflık tarayıcı, negatif test			

Kural no: GC-06			
Kural tanımı: Uygulama, yetki onaylama hizmetlerinin (LDAP, Active Directory) enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.			
Referans: OWASP ASVS 3.0 5.11	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: LDAP saldırısında kullanılan özel karakterlerin engellenmemesi (LDAP enjeksiyonu)	Zayıflık referansı: CWE-90	3	
Test yöntemi: Kaynak kodu açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod gözden geçirme			

Kural no: GC-07			
Kural tanımı: Uygulama, işletim sistemi komut enjeksiyonu açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.			
Referans: OWASP ASVS 3.0 5.12	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: İşletim sistemi komut saldırılarında kullanılan özel karakterlerin engellenmemesi ('İşletim sistemi komut enjeksiyonu')	Zayıflık referansı: CWE-78	3	
Test yöntemi: Kaynak kodu açıklık tarayıcısı, Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod gözden geçirme			

Kural no: GC-08			
Kural tanımı: Uygulama, girdi alınan içerik bir dosya yolu içeriyorsa, uzak ya da yakın dosya içerme açıklıklarını önleyici güvenlik denetimlerini yapmalıdır.			
Referans: OWASP ASVS 3.0 5.13	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Güvenilmeyen alandan işlevsellik içerme, Uzaktan dosya içerme	Zayıflık referansı: CWE-98, CWE-829, OWASP Top Ten 2004 Category A6 - Injection Flaws	3	
Test yöntemi: Kaynak kodu açıklık tarayıcısı, Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod gözden geçirme			

Kural no: GC-09			
Kural tanımı: Uygulama, XML açıklıklarını (XPath sorgu saldırıları, XML harici öge saldırıları, XML enjeksiyonu vb.) önleyici güvenlik denetimlerini yapmalıdır.			
Referans: OWASP ASVS 3.0 5.14	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: XML enjeksiyonu, XPath enjeksiyonu, XML harici öğelerinin yetersiz kısıtlanması	Zayıflık referansı: CWE-91, CWE-611, CWE-643	3	
Test yöntemi: Kaynak kodu açıklık tarayıcısı, Fuzz testi, Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod gözden geçirme			

Kural no: GC-10			
Kural tanımı: Eđer uygulamanın altyapısını oluŐturan teknoloji otomatik çoklu parametre birleŐtirme yeteneđine sahipse kötü amaçlı parametre ekleme saldırılarına karŐı koruma sađlanmalıdır.			
Referans: OWASP ASVS 3.0 5.16	Güvenlik Seviyesi:	2	3
KarŐı koyduđu zayıflıklar: Dinamik olarak belirlenen nesne özniteliklerinin kontrolsüz atanması	Zayıflık referansı: CWE-915		
Test yöntemi: İŐlevsel test, kaynak kod gözden geçirme, tasarım gözden geçirme			

Kural no: GC-11			
Kural tanımı: Uygulama, özellikle altyapı teknolojisi istek parametrelerinin kaynađı arasında bir ayırım yapmıyorsa, HTTP parametre kirliliđi saldırılarına karŐı savunma yapabilmelidir.			
Referans: OWASP ASVS 3.0 5.17	Güvenlik Seviyesi:	2	3
KarŐı koyduđu zayıflıklar: Fazla parametrelerin hatalı ele alınması, parametre enjeksiyonu veya ekleme, HTTP parametre kirliliđi	Zayıflık referansı: CWE-88, CWE-235, CAPEC-460		
Test yöntemi: Web uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test			

Kural no: GC-12			
Kural tanımı: HTML form alanlarının veri girdileri, REST çağrıları, HTTP üst başlıkları, çerezler, toplu işlem dosyaları, RSS beslemeleri gibi veri girdileri için doğrulama denetimi yapılmalıdır.			
Referans: OWASP ASVS 3.0 5.19	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Hatalı girdi doğrulama, Çıktının hatalı kodlanması	Zayıflık referansı: CWE-20, CWE-116		
Test yöntemi: Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod analizi, kaynak kod gözden geçirme			

Kural no: GC-14			
Kural tanımı: Yapısal olmayan veriler için izin verilen karakterler ve uzunluklar, verinin içeriđinde olabilecek olası zararlı karakterlerin denetlenmelidir.			
Referans: OWASP ASVS 3.0 5.21	Güvenlik Seviyesi:	1	2
KarŐı koyduđu zayıflıklar: Hatalı girdi doğrulama, Çıktının hatalı kodlanması	Zayıflık referansı: CWE-20, CWE-116		
Test yöntemi: Web uygulama açıklık tarayıcısı, uygulama açıklık tarayıcısı, tasarım gözden geçirme, işlevsel test, negatif test, kaynak kod analizi, kaynak kod gözden geçirme			

EK 2-Lahika-15: Web Servisleri Güvenliđi

Kural no: SG-01			
Kural tanımı: Uygulama, web servislerini iyi yapılandırılmış en az TLS v1.2 ve muadil güvenlik önlemi sunan bir protokol ile sunacak şekilde tasarlanmalıdır.			
Referans: Web Service Security Cheat Sheet	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Hassas bilgilerin şifrlenmemesi	Zayıflık referansı: CWE-311	3	
Test yöntemi: Tasarım Gözden Geçirme, uygulama açıklık tarayıcısı			

Kural no: SG-02			
Kural tanımı: Uygulama, web servis kimlik doğrulama ve yetkilendirmesi için oturum temelli yapılar kullanacak şekilde tasarlanmalıdır.			
Referans: OWASP ASVS 3.0 18.6	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Yetersiz kimlik doğrulama	Zayıflık referansı: CWE-287, CWE-285	3	
Test yöntemi: Tasarım Gözden Geçirme, kaynak kod gözden geçirme, uygulama açıklık tarayıcısı			

Kural no: SG-03			
Kural tanımı: Uygulamanın sunduđu SOAP temelli web servisleri en az (Web Services-Interoperability Basic Profile v1.0 ve üstü) ile uyumlu olacak şekilde tasarlanmalıdır.			
Referans: OWASP ASVS 3.0 18.5	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Standarda uygun güvenlik denetimi eksikliği	Zayıflık referansı: CWE-358	3	
Test yöntemi: Statik analiz, kaynak kodu kalite analizi aracı			

Kural no: SG-04			
Kural tanımı: Uygulama, web servis yapılandırma ve yönetim işlevlerine sadece yetkili kullanıcıların erişebilmesini sağlamalıdır.			
Referans: OWASP ASVS 3.0 18.2	Güvenlik Seviyesi:	1	2
Karşı koyduđu zayıflıklar: Kaynak yönetiminde risk oluşturan işlemler	Zayıflık referansı: CWE-752	3	
Test yöntemi: Gözden Geçirme, uygulama açıklık tarayıcısı			

Kural no: SG-05			
Kural tanımı: Uygulama, web servis girdilerini kullanmadan önce gidilerin şeklini (XML ve JSON şemalarına uygunluk, parametre beyaz listesi) uygunluđunu ve içeriđini çeşitli saldırılara karşı (XML bombalama, dış varlık saldırısı, kusurlu XML yapısı, tekrarlamalı girdi vb.) kontrol etmelidir.			
Referans: OWASP ASVS 3.0 18.3	Güvenlik Seviyesi:	1	2
		3	

Karşı koyduđu zayıflıklar: Yetersiz girdi dođrulama	Zayıflık referansı: CWE-20
Test yöntemi: uygulama açıklık aracı, kaynak kodu kalite analizi aracı	

Kural no: SG-06				
Kural tanımı: Uygulama, sunucu ve istemci tarafında dil kodlaması (encoding) saldırılarına karşı dayanıklı olmalıdır.				
Referans: OWASP ASVS 3.0 18.1	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Veri kodlaması hatası	Zayıflık referansı: CWE-172			
Test yöntemi: uygulama açıklık aracı, kaynak kodu kalite analizi aracı				

Kural no: SG-07				
Kural tanımı: Uygulama, web servisi ile gönderilen veride betik (script) içermeyecek şekilde tasarlanmalıdır.				
Referans: Web Service Security Cheat Sheet	Güvenlik Seviyesi:			3
Karşı koyduđu zayıflıklar: Çıktının hatalı kodlanması	Zayıflık referansı: CWE-116			
Test yöntemi: Statik analiz, kaynak kodu kalite analizi aracı				

Kural no: SG-08				
Kural tanımı: Uygulama, web servislerinden şifreli olarak paylaşılan verileri yine şifreli olarak saklayacak şekilde tasarlanmalıdır.				
Referans: Web Service Security Cheat Sheet	Güvenlik Seviyesi:	1	2	3
Karşı koyduđu zayıflıklar: Hassas bilgilerin şifrelenmemesi	Zayıflık referansı: CWE-311			
Test yöntemi: Tasarım Gözden Geçirme, uygulama açıklık tarayıcısı				

EK 2-Lahika-16: Kişisel Verilerin Korunması

Kural no: KV-01			
Kural tanımı: Uygulama, kişisel veriler üzerinde işlem yapılması ana amaç olmayan durumlarda kişisel verileri maskeleyerek görüntülemeli, aktarmalı veya işlemelidir.			
Referans:	Güvenlik Seviyesi:	1	2
Karşı koyduğu zayıflıklar: Mahremiyet ihlali	Zayıflık referansı: CWE-359	3	
Test yöntemi: Gözden geçirme			

Kural no: KV-02			
Kural tanımı: Uygulama, kişisel verileri şifreli olarak saklamalı ve bu verilerin taşınmasında korumalı iletişim kanallarını kullanmalıdır.			
Referans: OWASP ASVS 3.1 7 4	Güvenlik Seviyesi:	2	3
Karşı koyduğu zayıflıklar: Yazılım hassas bilgileri, yetkisiz kişilerce okuma veya yazma erişimini sınırlamadan saklar.	Zayıflık referansı: CWE-922, CWE-921, CWE-312, CWE-319		
Test yöntemi: Tasarım Gözden Geçirme, Statik kod analizi, gözden geçirme			

Kural no: KV-03			
Kural tanımı: Uygulamanın istemci tarafında çalışan kodları, kişisel verileri başka ortamlara aktarmamalı (konsola yazma, başka dosya olarak kaydetme, yerel veya uzak uygulamalara transfer etme vb.), güvensiz ortamlarda (ortak dizin, USB disk vb.) güvensiz yöntemlerle (açık metin olarak, zayıf şifreleme algoritma kullanarak şifreleme vb.) saklamamalıdır.			
Referans: OWASP ASVS 3.0 1 10	Güvenlik Seviyesi:	2	3
Karşı koyduğu zayıflıklar: Koda gömülü kimlik bilgilerini kullanma, koda gömülü güvenlik sabitleri kullanma, kaynak kodu yoluyla bilgi sızması	Zayıflık referansı: CWE-798, CWE-547, CWE-540		
Test yöntemi: Statik analiz, kaynak kodu kalite analizi aracı			

Kural no: KV-04			
Kural tanımı: Kullanılan veritabanının dışarıya aktarımı ancak veritabanı yönetim yetkisi olan hesaplarla yapılmalı ve öncesinde veritabanındaki kişisel verilerin silinmesi sağlanmalıdır.			
Referans: Appsec STIG SV-85033r1_rule	Güvenlik Seviyesi:	2	3
Karşı koyduğu zayıflıklar: Hassas verinin sınırda uygun şekilde kaldırılamaması	Zayıflık referansı: CWE-212		
Test yöntemi: Gözden geçirme, işlevsel test			