İSTANBUL MEDENİYET ÜNİVERSİTESİ

2025-2026 AKADEMİK YILI GÜZ DÖNEMİ

BİL403 YAZILIM MÜHENDİSLİĞİ

9. DERS

Dr. Öğr. Üyesi Ertürk ERDAĞI
erturk.erdagi@medeniyet.edu.tr

Yazılım Güvenliği Nedir?

- Yazılım güvenliği, bir yazılıma yönelik olası saldırıların, hataların ve istenmeyen davranışların önlenmesi için alınan teknik ve süreç temelli önlemler bütünüdür. Amaç:
- Sistemin **gizliliğini,**
- **bütünlüğünü,**
- **erişilebilirliğini,**
- **hesap verebilirliğini,**
- **dayanıklılığını** korumaktır.
- Modern yazılımlar, birçok bileşen, servis ve API'den oluşan karmaşık yapılardır.
- Her bileşen olası bir saldırı yüzeyidir.
- CVE kayıtları: 2023 yılında 29.000'den fazla yeni zafiyet (tarihteki en yüksek sayı).
- Güvenlik açıkları → finansal kayıp, veri sızıntısı, sistem kesintisi, itibar kaybı.

Yazılım Güvenliğinde Temel Kavramlar

- Güvenli yazılımın üç temel hedefi: CIA

Kavram	Açıklama	Örnek
Confidentiality (Gizlilik)	Yetkisiz kişilerin erişimini engellemek	Veritabanı şifresinin açıkta tutulmaması
Integrity (Bütünlük)	Verinin doğruluğunun bozulmaması	Banka havalesinde miktarın değiştirilmemesi
Availability (Erişilebilirlik)	Sistemin hizmet verebilir durumda kalması	DDOS saldırısına dayanıklı sunucu

Güvenlikle İlgili Temel Terimler

- Vulnerability (Zafiyet) : Sistemdeki bir güvenlik açığı.
- Threat (Tehdit) : Sisteme zarar verme potansiyeli olan durum.
- Attack (Saldırı) : Tehdidin uygulanması.
- Exploit : Açığı kötüye kullanan araç ya da kod.
- Attack Surface (Saldırı Yüzeyi) : Saldırganın girdi sağlayabileceği tüm yerler.
- Security Policy : Organizasyonun güvenlik kuralları.
- Authentication : Kimlik doğrulama (kimsin?).
- Authorization : Yetki kontrolü (ne yapabilirsin?).
- Auditing & Logging : İzlenebilirlik ve hesap verebilirlik.

Güvenlik Açığı Nasıl Oluşur?

- Hatalı tasarım
- Yanlış yapılandırma (config)
- Zayıf doğrulama
- Güncel olmayan bağımlılıklar
- Kötü kodlama pratikleri
- Yetersiz test
- İnsan hatası
- Zayıf parola politikaları
- Kötü mimari kararlar

Güvenlik Gereksinimleri ve Yazılım Mühendisliği Bağlantısı

- ISO/IEC 25010 kalite modeline göre güvenlik; yazılım kalitesinin alt özniteliklerinden biridir.
- Alt başlıklar: Gizlilik, Bütünlük, Hesap verebilirlik, Yetkilendirme, Dayanıklılık
- Önceki derslerle bağ:
  - "Yazılım Kalitesi" dersinde anlattığımız kalite öznitelikleri içinde güvenlik kritik bir bileşendir.
- **Gereksinim Mühendisliğinde Güvenlik Gereksinimleri**
  - Fonksiyonel olmayan gereksinimler içinde yer alır.
  - Gereksinimler net ve ölçülebilir olmalıdır.
- **Örnek kötü gereksinim**
  - "Sistem güvenli olmalıdır." Bu gereksinim *anlamsızdır*, ölçülemez.
- **Örnek iyi gereksinim**
  - "Tüm parolalar en az 12 karakter olacak ve bcrypt ile saklanacaktır."
  - "Sistem, başarısız giriş denemelerinde 5 hatalı deneme sonrası hesap kilitleme sağlayacaktır."

PRISM Güvenlik Gereksinimleri Modeli

- **PRISM modeli**, yazılım güvenliği gereksinimlerini sistematik bir şekilde analiz etmek için kullanılan çok boyutlu bir çerçevedir. Model, güvenliği sadece "teknik bir özellik" olarak değil; sistemin korunması gereken misyonu, insan faktörü, veri gizliliği ve operasyonel güvenilirliği ile birlikte ele alır.
- PRISM modeli güvenliği 5 boyutta ele alır. Bu boyutlar hem **güvenlik gereksinimlerinin tanımlanmasında**, hem de **tehdit modelleme ve risk analizinde** kullanılır.

Boyut	Açıklama
Privacy	Kullanıcı bilgilerinin korunması
Risk	Tehditlerin etkisi ve olasılığı
Integrity	Verinin bozulmaya karşı dirençli olması
Safety	Sistemin insan/hukuk açısından güvenli olması
Mission Assurance	Kritik sistemlerde kesintisiz çalışabilirlik

PRISM Güvenlik Gereksinimleri Modeli - Privacy

- Sistemde işlenen verilerin, yalnızca yetkili kişiler tarafından erişilebilir kalmasını sağlayan gereksinimler bütünüdür.
- **Neyi Kapsar?**
  - Kişisel verilerin korunması
  - Erişim kontrolü
  - Veri sınıflandırma politikaları
  - Kriptografi
  - GDPR / KVKK gibi yasal gereksinimler
- **Örnek Güvenlik Gereksinimleri**
  - "Kullanıcı parolaları bcrypt ile hash'lenmiş olarak depolanacaktır."
  - "Tüm veri iletimi TLS 1.3 protokolü üzerinden yapılacaktır."
  - "Öğrenci bilgileri yalnızca yönetici rolüne sahip kullanıcılar tarafından görüntülenebilir."
- **Tipik Saldırı Türleri**
  - Information disclosure
  - SQL injection → gizli veriye erişim
  - Broken Access Control
  - Zayıf şifreleme
- **CIA Triadı ile ilişkisi:** Confidentiality

PRISM Güvenlik Gereksinimleri Modeli - Risk

- Sisteme yönelik tehditlerin olasılığı ve etkisinin değerlendirilerek önceliklendirilmesi.
- **Neyi Kapsar?**
  - STRIDE tehdit analizi
  - DREAD risk skoru
  - Güvenlik gereksinimlerinin önceliklendirilmesi
  - Olasılık ⟶ Etki ⟶ Önceliklendirme döngüsü
- **Risk Yönetimi Sorusunun Mantığı**
  - “Bu tehdit gerçekleşirse ne olur?
  - Ne kadar zarar verir?
  - Gerçekleşme ihtimali nedir?”
- Bu boyut; güvenlik gereksinimlerinin proje bütçesine, süreye ve teknik kapasiteye göre optimize edilmesini sağlar.
- **Örnek Güvenlik Gereksinimleri**
  - “Kritik hata logları gerçek zamanlı olarak SIEM sistemine iletilecektir.”
  - “Yüksek riskli zafiyetler (CVSS ≥ 7.0) 48 saat içinde yamalanacaktır.”

PRISM Güvenlik Gereksinimleri Modeli - Integrity (Bütünlük)

- Veri ve sistem bileşenlerinin **yetkisiz değiştirilememesi**, **bozulmaması** ve **tutarlılığını kaybetmemesi**.
- **Neyi Kapsar?**
  - Veri bütünlüğü
  - Kod bütünlüğü
  - Dosya imzalama
  - Hash mekanizmaları
  - Logların değiştirilemez olması
- **Örnek Güvenlik Gereksinimleri**
  - "Veritabanı güncellemeleri transaction log ile kaydedilecek, geri alınabilir olmalıdır."
  - "Kritik uygulama dosyaları SHA-256 ile bütünlük doğrulamasından geçecektir."
  - "Log dosyalarına müdahale edilirse sistem yöneticisine uyarı gönderilecektir."
- **CIA Triadı ile ilişkisi:** Integrity
- **Tipik Saldırı Türleri**
  - Tampering
  - Man-in-the-middle
  - Hash collision manipülasyonları
  - Yetkisiz veri güncelleme

PRISM Güvenlik Gereksinimleri Modeli - Safety (Emniyet/Güvenlik)

- Sistemin insanlara, çevreye veya organizasyona zarar vermemesini sağlayan gereksinimler. Bu boyut özellikle:
- Otonom sistemler
  - IoT
  - Endüstriyel kontrol sistemleri
  - Medikal cihaz yazılımları
  - Ulaşım sistemleri
    gibi kritik uygulamalarda önemlidir.
- **Safety Neyi Kapsar?**
  - Hatalı davranışın fiziksel zarara neden olmaması
  - Yasal uyumluluk
  - Güvenlik-kritik fonksiyonların kontrolü
  - Fail-safe tasarım
  - Acil durum mekanizmaları
- **Örnek Güvenlik Gereksinimleri**
  - “Sensör hatası durumunda araç otomatik olarak güvenli moda geçecektir.”
  - “Medikal cihaz yazılımı, kritik dozaj hatalarını engellemek için ikinci kontrol mekanizmasına sahip olacaktır.”
  - “Kullanıcıdan gelen hatalı komutlar doğrulanmadan işlenmeyecektir.”

PRISM Güvenlik Gereksinimleri Modeli - Mission Assurance (Görev Güvencesi)

- Bu boyut, sistemin **zorlu koşullarda bile görevini sürdürebilmesi** ile ilgilidir. Sistemin kritik işlevlerini saldırılar, hatalar ve beklenmedik durumlar karşısında sürdürebilmesini sağlayan gereksinimler.
- **Neyi Kapsar?**
  - Yüksek erişilebilirlik
  - Redundancy (yedeklilik)
  - Failover mekanizmaları
  - Dayanıklılık (resilience)
  - Güvenlik olayı sonrası toparlanma (recovery)
  - İş sürekliliği planları
- **Örnek Güvenlik Gereksinimleri**
  - “Sistem, bir sunucu arızasında 5 saniye içinde yedek sunucuya geçiş yapacaktır.”
  - “Tüm kritik veriler günlük olarak off-site yedeklenecektir.”
  - “Ransomware saldırısı tespit edildiğinde dosya sistemi salt-okunur moda alınacaktır.”
- **CIA Triadı ile ilişkisi:** Availability

PRISM Modelinin Yazılım Mühendisliği Açısından Önemi

Aşama	PRISM'in Rolü
Gereksinim Analizi	Güvenlik gereksinimleri net, ölçülebilir ve kategorize edilerek yazılır
Tasarım	Privacy, Integrity ve Safety yüksek seviyeli mimariyi belirler
Uygulama	Risk yönetimi ve bütünlük kontrolleri kodlama standartlarını etkiler
Test	PRISM boyutları test stratejisini belirler
Bakım	Mission assurance → yamalama, güncelleme, izleme süreçleri

Tehdit Modelleme Nedir?

- Tehdit modelleme, bir yazılım sistemini *potansiyel saldırgan gibi düşünerek* olası tehditleri, zafiyetleri ve saldırı yüzeylerini sistematik biçimde analiz etme sürecidir.
- **Neden Tehdit Modelleme Yapılır?**
  - Sistem saldırıya uğramadan önce zayıf noktaları belirlemek
  - Güvenlik gereksinimlerini ortaya çıkarmak
  - Tasarım aşamasında güvenlik açıklarını tespit etmek
  - Riskleri teknik olmayan paydaşlara açıklamak
  - Sızma testine hazırlık yapmak
  - Güvenlik bütçesini doğru kullanmak
- **Tehdit Modelleme Ne Zaman Yapılır?**
  - Gereksinim belirleme
  - Sistem tasarımı
  - Kod geliştirme
  - Yayın sonrası değişikliklerde

Tehdit Modelleme Süreci

Adım	Açıklama
1. Varlıkları Tanımla (Assets)	Korunması gereken şey nedir? (Veri, kullanıcı, hizmet)
2. Roller ve Saldırgan Profilleri	Kim saldırabilir? (insider, outsider, script kiddie, APT)
3. Saldırı Yüzeyini Belirle	Giriş noktaları, API uçları, form alanları, ağ arayüzleri
4. Tehditleri Belirle	STRIDE veya diğer tehdit sınıfları kullanılır
5. Risk Analizi Yap	DREAD / CVSS / etki-olasılık değerlendirmesi
6. Karşı Önlemleri Tanımla	Güvenlik gereksinimlerine dönüştür

STRIDE Tehdit Modeli

- **STRIDE**, yazılım tasarımında tehditleri sınıflandırmak için kullanılan, en yaygın modellerden biridir.

Harf	Tehdit Türü	Açıklama	Örnek
S	Spoofing	Kimlik sahteciliği	Başkasının hesabına giriş yapma
T	Tampering	Veri manipülasyonu	Banka transfer tutarını değiştirme
R	Repudiation	İnkar edilebilirlik	Log'ların tutulmaması
I	Information Disclosure	Hassas veri sızıntısı	Parola, TC kimlik açıkta
D	Denial of Service	Hizmet engelleme	DDOS saldırısı
E	Elevation of Privilege	Yetki yükseltme	Normal kullanıcı → admin olma

Risk Analizi – DREAD Modeli

- DREAD, tehditlerin risk seviyesini hesaplamak için kullanılan bir çerçevedir.

Harf	Açılım	Anlamı
D	Damage	Ne kadar zarar verir?
R	Reproducibility	Saldırı tekrarlanabilir mi?
E	Exploitability	Ne kadar kolay istismar edilir?
A	Affected Users	Kaç kullanıcı etkilenir?
D	Discoverability	Açık ne kadar kolay keşfedilir?

Secure Software Development Life Cycle (SSDLC)

- SSDLC (**Secure Software Development Life Cycle**), güvenlik gereksinimlerinin yazılım yaşam döngüsünün **en başından itibaren** ele alınmasını sağlayan yöntemdir.
- **Neden SSDLC?**
  - Güvenlik açıklarının %70'i tasarım ve kodlama aşamasında ortaya çıkar.
  - Yayın sonrası düzeltmek, tasarım aşamasında önlem almaktan **30 kat daha maliyetlidir**.
  - Regülasyonlar (KVKK, GDPR, ISO 27001) erken güvenlik ister.
  - Yazılım kalitesinin ve güvenilirliğin artması.
- **Geleneksel SDLC'de eksik olan ne?**
  - Güvenlik genellikle test aşamasına bırakılır → **çok geç**
  - Tehdit modellemesi yapılmaz
  - Kod incelemeleri güvenlik odağında değildir
  - Bileşen güncellemeleri dikkate alınmaz

Microsoft Security Development Lifecycle (MS SDL) Yaklaşımı

- **Microsoft** tarafından kurumsal projelerde kullanılan SSDLC modelidir.
- **Aşamalar:**
- **Education (Eğitim) :** Tüm ekip üyelerinin temel güvenlik farkındalığına sahip olması.
- **Requirements (Gereksinimler) :** Sistemin güvenlik ihtiyaçlarını açıkça tanımlamak.
- **Design (Tasarım) :** Bu aşamada tasarım kararları güvenlik göz önünde bulundurularak alınır.
- **Implementation (Kodlama):** Güvenli kod geliştirme
- **Verification (Doğrulama/Test) :** Güvenlik testleri
- **Release (Yayın) :** Bu aşamada güvenlik ile ilgili yapılması gerekenler tamamlanmalı
- **Response (Bakım/İzleme) :** Güvenlik İzleme, Olay Yönetimi

DevOps

- **DevOps**, *Development (Yazılım Geliştirme)* ve *Operations (Sistem/Operasyon Yönetimi)* ekipleri arasındaki iletişimi, işbirliğini ve otomasyonu artırmayı amaçlayan bir kültür, felsefe ve pratikler bütünüdür.
- Yazılımın geliştirilmesi → test edilmesi → dağıtılması → çalıştırılması → izlenmesi **tek, bütünleşik bir süreç** haline gelir.
- DevOps bir **araç** değildir;
  - bir **kültür**,
  - bir **yaklaşım**,
  - bir **işbirliği modelidir**.
- Geleneksel modelde:
  - Geliştirici "kod çalışıyor" der,
  - Operasyon ekibi "sunucu çalışmıyor" der.
- Bu çatışmayı ortadan kaldıran model → **DevOps**
- **DevOps'un temel prensibi:**
  - "Yazılım geliştirme, test, dağıtım ve işletim *ayrı ekiplerin* değil, *tek bir bütünsel sürecin* parçalarıdır."

DevOps'un Temel Hedefleri

- DevOps Neden Gereklidir?
- **Sorun: Geleneksel Süreç**
  - Geliştirici kodu yazar → işi biter.
  - Operasyon ekibi sunucuyu yönetir → sorumluluk onlardadır.
  - Hatalar geç fark edilir.
  - Yayınlama yavaş ve risklidir.
- **Çözüm: DevOps**
  - Ekipler birlikte çalışır.
  - Dağıtım hattı otomatikleştirilir.
  - Güvenlik, test ve dağıtım sürecin içine gömülür.
  - Hata → dakika içinde bulunur.
  - Yeni sürüm → her an yayınlanabilir hale gelir.

Hedef	Açıklama
Sürekli Entegrasyon (CI)	Kodların sık sık otomatik olarak birleştirilmesi
Sürekli Dağıtım / Sürekli Teslimat (CD)	Yazılımın güvenli ve otomatik şekilde yayınlanması
Otomasyon	Testlerin, build sürecinin, dağıtımın otomatik hale gelmesi
İşbirliği ve Kültür	Geliştirici–operasyon ekiplerinin ortak sorumluluk alması
Hızlı geri bildirim	Hataların erken bulunması ve hızlı çözülmesi
İzleme ve geri kazanım	Üretim ortamının sürekli izlenmesi

SSDLC ve Agile / DevOps İlişkisi

- ✔ **Agile'de Güvenlik**
  - Her sprint'te güvenlik testleri
  - "Security as a Task"
  - Sprint bazlı tehdit modelleme
- ✔ **DevSecOps**
  - DevOps'un güvenlik eklenmiş halidir. DevOps pipeline'a:
  - Static code analysis
  - Container güvenliği
  - Vulnerability scanning entegre edilir.

OWASP Nedir?

- OWASP (**Open Web Application Security Project**), web uygulaması güvenliği alanında dünyanın en büyük topluluğudur.
- Her 3 yılda bir, web uygulamalarında en sık görülen kritik riskleri listeleyen **OWASP Top 10** belgesini yayınlar.
- **Amaç:**
  - Geliştiricilere, testçilere ve mimarlara “en kritik güvenlik açıklarının” farkındalığını kazandırmak.
  - Güvenli yazılım geliştirme standartları oluşturmak.

OWASP Top 10

- **A01 – Broken Access Control (Bozuk Erişim Kontrolü)**
  - Sistem, kullanıcıların yetkilerini doğru kontrol etmezse ortaya çıkar.
  - Kullanıcılar yetkili olmadıkları verilere erişebilir veya işlemler yapabilir.
  - Kullanıcının URL değiştirerek başka bir hesabın verisine erişmesi: **/user/profile?id=123 → 124**
  - Korunma Yöntemleri : Rol tabanlı erişim kontrolü, yetki kontrolünün *sunucu tarafında* yapılması
- **A02 – Cryptographic Failures (Kriptografi Hataları)**
  - Verilerin yanlış şifrelenmesi, yanlış algoritmaların kullanılması veya hiç şifreleme yapılmaması.
  - Korunma Yöntemleri : Modern algoritmalar, HTTPS zorunluluğu
- **A03 – Injection (Enjeksiyon Saldırıları)**
  - Kullanıcı girdisinin doğrulanmadan bir komuta, sorguya veya şablona enjekte edilmesi.
- **A04 – Insecure Design (Güvensiz Tasarım)**
  - Kod yazılmadan önce yapılan tasarım hataları.
- **A05 – Security Misconfiguration (Güvenlik Yapılandırma Hataları)**
  - Sistem yapılandırmalarının yanlış veya eksik yapılmasından kaynaklanan açıklar.
  - Örnek : Default şifrelerin değiştirilmemesi,

OWASP Top 10

- **A06 – Vulnerable and Outdated Components (Güncel Olmayan Bileşenler)**
  - Kütüphaneler, framework'ler, paketler güncel değilse saldırganlar bilinen zafiyetleri kullanabilir.
- **A07 – Identification and Authentication Failures (Kimlik Doğrulama Hataları)**
  - Örnek : zayıf parola politikası
- **A08 – Software and Data Integrity Failures (Yazılım ve Veri Bütünlüğü Hataları)**
  - Bir bileşenin veya verinin manipüle edilmesi.
- **A09 – Security Logging and Monitoring Failures (Loglama ve İzleme Hataları)**
  - Güvenlik olaylarının loglanmaması veya izlenmemesi.
- **A10 – Server-Side Request Forgery (SSRF)**
  - Saldırganın sunucuyu, başka bir sunucuya istek göndermeye zorlaması.

Uygulamalı Güvenlik Testleri

- **1.Statik Uygulama Güvenlik Testi**
- SAST, uygulama **çalıştırılmadan**, kodun analiz edilerek güvenlik açıklarının tespit edilmesidir.
- **Kod seviyesinde erken tespit** sağlar.
- **Avantajları**
  - Kod çalışmadan açıkları bulur
  - Build pipeline'a (CI) kolay entegre edilir
  - Otomatik tarama yapılabilir
  - Erken aşamada düzeltme → düşük maliyet
- **Dezavantajları**
  - False positive (yanlış alarm) verebilir
  - Kodun tamamını anlamaz, bağlam eksik olabilir

Uygulamalı Güvenlik Testleri

- **2. Dinamik Uygulama Güvenlik Testi**
- DAST, uygulama çalışırken dışarıdan saldırgan bakış açısıyla test yapar.
- Siyah kutu testine benzer.
- Web uygulamalarında çok yaygındır.
- **Avantajlar**
  - Gerçek çalışma ortamını test eder
  - Kullanıcı giriş noktalarını analiz eder
  - Otomatik saldırı simülasyonları çalıştırır
- **Dezavantajlar**
  - Kaynak kodunu analiz etmez
  - Bazı hataları göremez (örn: kod içi mantık açıkları)

Uygulamalı Güvenlik Testleri

- **3. Pentest**
- Pentest, saldırgan davranışını taklit ederek sistemin zayıflıklarını ortaya çıkaran derinlemesine güvenlik testidir.
- **Pentest Aşamaları**
- **Keşif (Reconnaissance)**
  - Domain, IP, açık servisler
- **Taramalar (Scanning)**
  - Port scan
  - Nmap
- **Zafiyet Analizi**
  - Nessus, OpenVAS
- **Saldırı / Exploitation**
  - Exploit çalıştırma (Metasploit vb.)
- **Yetki Yükseltme (Privilege Escalation)**
- **Erişim Kalıcılığı**
- **İz Temizleme**
- **Raporlama**

Test Türü	Açıklama
Black-box	Saldırgan hiçbir bilgiye sahip değil
White-box	Tüm kaynak kodu ve bilgiler mevcut
Grey-box	Sınırlı bilgi ile test

Örnek Pentest Senaryosu

Bir login formunda:

•Sıfırlama linki tahmin edilebiliyor

•Parola brute-force denemesi engellenmemiş

•Captcha yok

•Hata mesajları gereğinden fazla bilgi veriyor

Pentest sonucu:

⟶ Authentication bypass

⟶ Account takeover riski

Güvenli Veri Yönetimi ve Kriptografi Temelleri

- Kriptografi, veriyi yetkisiz erişime karşı korumak için matematiksel yöntemlerin kullanılmasını sağlar.
- **Kriptografinin Amaçları (CIA + 2)**
  - Gizlilik (Confidentiality)
  - Bütünlük (Integrity)
  - Availability (Erişilebilirlik)
  - Yetkilendirme (Authorization)
  - İnkar Edilemezlik (Non-repudiation)
- **Hash Nedir?** : Verinin tek yönlü bir matematiksel fonksiyonla özetlenmesidir.
  - Geri döndürülemez.
  - Aynı veri → her zaman aynı hash.
  - **Kullanım Alanları :** Parola saklama, Dosya bütünlüğü doğrulama, Veri manipülasyonunun tespiti

İSTANBUL MEDENİYET ÜNİVERSİTESİ

KATILIMINIZ İÇİN TEŞEKKÜR EDERİM.

YOKLAMA İÇİN İMZANIZI ATMAYI UNUTMAYINIZ.
CLASSROOM ÜZERİNDEN SINIFA DAHİL OLMAYI UNUTMAYINIZ

Sınıf Kodu : pua2tnoe

Dr. Öğr. Üyesi Ertürk ERDAĞI
erturk.erdagi@medeniyet.edu.tr