# OWASP TOP 10 ARAŞTIRMASI

## 1. Broken Access Control

- ➢ Kullanıcıların yetkili izinlerini aşmaaları durumunda yetkisiz bilgi ifşasına, veri değişikliğine veya iş işlevlerini sınırlarının ötesinde gerçekleştirmelerine neden olabilecek bir zafiyettir.
- ➢ Kullanıcı doğrulama mekanizması yeterince doğru hazırlanmamışsa bu zafiyete yol açar.
- ➢ Bu zafiyetin türleri Dikey Ayrıcalık Yükseltme, Yatay Ayrıcalık Yükseltme, Çok Adımlı İşlemlerde Erişim Kontrolü Zafiyetleridir.
- ➢ **Bu zafiyeti önlemek için :**
  - ○ Güçlü kimlik doğrulama ve oturum yönetimi kullanın.
  - ○ Erişim kontrollerini düzenli olarak test edin ve güncelleyin.
  - ○ Güvenli kodlama uygulamaları kullanın.
  - ○ Güvenlik güncellemelerini izleyin ve uygulayın.

## 2. Cryptographic Failures

- ➢ Kriptografi kullanılarak şifrelenen dosyaların şifrelerinin çözülmesi veya doğru şekilde şifrelenmemesi sonucunda korunmak istenen verilerin sızdırılmasına yol açabilecek zafiyettir.
- ➢ Eskimiş veya zayıf şifreleme algoritmalarının kullanılması, TLS kullanmaması gibi türleri vardır.
- ➢ **Bu zafiyeti önlemek için:**
  - ○ Tüm hassas verileri şifrelediğinizden emin olun.
  - ○ Hassas verigeri gerekmedikçe saklamayın.
  - ○ Şifrelerken tahmin edilebilir tuzlama yöntemi kullanmayın. Rastgele oluşturun.
  - ○ Hassas verilerin taşınması için FTP ve SMTP gibi eski protokolleri kullanmayın.

## 3. Injection

- ➢ Bu güvenlik açığı, kötü niyetli kullanıcıların veri girdilerinin yanlış kontrol edilmesi veya filtrelenmesi nedeniyle uygulama tarafından yürütülen veritabanı sorgularına, komutlarına veya diğer işlevlere kötü amaçlı kod enjekte etmesine olanak tanır.
- ➢ SQL Injection, Command Injection, NoSQL Injection gibi türleri vardır.
  - ○ SQL Injection, bir saldırganın bir uygulamanın veritabanına yaptığı sorgulara müdahale etmesine olanak tanıyan bir web güvenlik açığıdır.
  - ○ Command injection zafiyeti kullanıcının girmiş olduğu verinin yanında Linux ya da windows komutları çalıştırmasına olanak sağlar.
  - ○ NoSQL Injection SQL Injection gibi saldırganların kimlik doğrulamalarını atlayıp saklı verilere erişebilmelerini sağlar. Ama veriler SQL'den farklı bir şekilde saklandığı için zafiyetin uygulanması da farklıdır.
- ➢ **Bu zafiyeti önlemek için:**
  - ○ Özel karakerlerin kullanımını sınırlayabilirsiniz.
  - ○ WAF Kullanabilisiniz.

## 4. Insecure Design

➢ Insecure Design, bir web uygulamasının tasarımında yapılan hatalar veya eksiklikler nedeniyle ortaya çıkan bir güvenlik açığıdır. Insecure Design, uygulamanın özellikle kimlik doğrulama, yetkilendirme, veri gizliliği ve bütünlüğü gibi önemli güvenlik konularında hatalar içermesiyle ortaya çıkabilir

➢ **Bu zafiyeti önlemek için:**

- Güvenli tasarım ilkeleri uygulanmalıdır.
- Güvenlik açıkları düzeltin.
- Kaynak tüketimini kullanıcı veya hizmete göre sınırlayın.
- Kritik kimlik doğrulama, erişim kontrolü, iş mantığı ve anahtar akışları için tehdit modellemesini kullanın.

## 5. Security Misconfiguration

➢ Sunucu veya web uygulamalarında güvenlik kontrollerinin istendiği halde yanlış yapılandırması veya hiç yapılanmaması sonucunda ortaya çıkar.

➢ **Bu zafiyeti önlemek için:**

- Kullanılmayan özellikleri ve çerçeveleri kaldırın veya yüklemeyin.
- Güvenlik yamalarını ve güncellemelerini takip edin.

## 6. Vulnerable and Outdated Components

➢ Bir uygulamada daha önce açık bulunmuş üçüncü taraf yazılımların güncellenememesi veya bilinen güvenlik açıklarına sahip olması nedeniyle bu zafiyet ortaya çıkar.

➢ **Bu zafiyeti önlemek için:**

➢ Kullanılmayan özellikleri, bileşenleri, dosyaları ve belgeleri kaldırın.

➢ Güncelleme politikaları oluşturun.

## 7. Identification and Authentication Failures

➢ Kimlik doğrulaması ve oturum yönetimi sağlanmaması nedeniyle ortaya çıkan zafiyettir.

➢ **Bu zafiyeti önlemek için:**

- Mümkün olan yerlerde otomatik kimlik bilgisi doldurma, kaba kuvvet ve çalınan kimlik bilgilerinin yeniden kullanımı saldırılarını önlemek için çok faktörlü kimlik doğrulama uygulayın.
- Özellikle yönetici kullanıcılar için herhangi bir varsayılan kimlik bilgisi ile göndermeyin veya dağıtmayın.
- Çok faktörlü kimlik doğrulama yöntemlerini kullanmak

## 8. Software and Data Integrity Failures

➢ Yazılım ve veri bütünlüğü hataları, bütünlük ihlallerine karşı koruma sağlamayan kod ve altyapı ile ilgilidir. Bunun bir örneği, bir uygulamanın güvenilmeyen kaynaklardan, depolardan ve içerik dağıtım ağlarından gelen eklentilere, kütüphanelere veya modüllere dayanmasıdır.

➢ **Bu zafiyeti önlemek için:**

- Yazılımın veya verilerin beklenen kaynaktan geldiğini ve değiştirilmediğini doğrulamak için dijital imzalar veya benzer mekanizmalar kullanın.

- Npm veya Maven gibi kütüphanelerin ve bağımlılıkların güvenilir depoları kullandığından emin olun. Daha yüksek bir risk profiliniz varsa, incelenmiş, iyi bilinen dahili bir depo barındırmayı düşünün.

## 9. Security Logging and Monitoring Failures

➢ Security Logging and Monitoring Failures, güvenlik olaylarının izlenmesi ve kaydedilmesi işlevlerinin yetersizliği veya hatalı yapılandırılması sonucu ortaya çıkan bir güvenlik açığıdır.

➢ **Bu zafiyeti önlemek için:**

- Güvenlik olaylarının izlenmesi ve kaydedilmesi için uygun araçlar kullanmak
- Günlük kayıtlarının düzenli olarak incelenmesi
- Uyarı ve alarm sistemleri kullanmak

## 10. Server-Side Request Forgery

➢ Server-Side Request Forgery,bir saldırganın hedef sunucuda bir URL'ye istek gönderirken sahte bir kaynak IP adresi ve alan adı sağlayarak sunucunun kendi iç ağlarına veya diğer harici kaynaklara erişmesine izin veren bir web güvenliği açığıdır.

➢ **Bu zafiyeti önlemek için:**

- Giriş doğrulaması
- Güvenlik duvarı kurulumu
- Güvenli URL işleme
- Sunucu ayarlarının kontrol edilmesi