YAVUZLAR WEB GÜVENLİĞİ & YAZILIM TAKIMI
OWASP TOP 10 ÖDEV RAPORU

1. Zafiyet Nedir

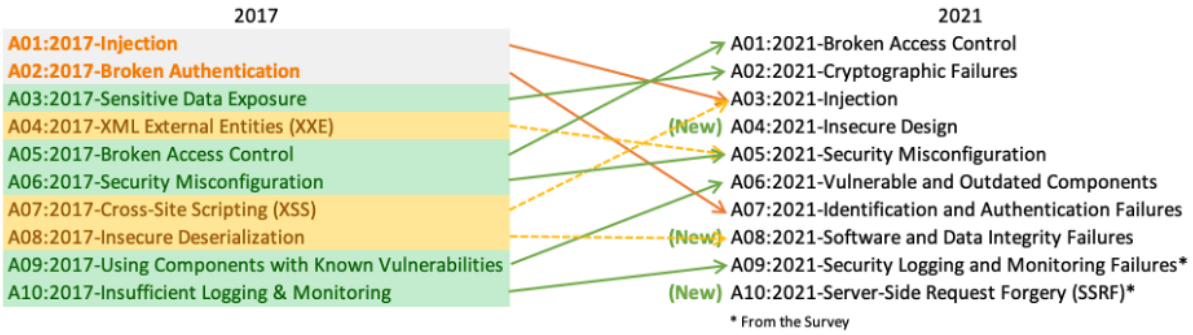
Web uygulamaları için zafiyet, web tabanlı bir uygulamadaki, saldırganların uygulamanın güvenliğini tehlikeye atmak için kullanabileceği bir kusur veya zayıflığı ifade eder. İnternetin global erişime sahip olması nedeniyle, web uygulaması zafiyetleri saldırılara karşı özellikle hassastır ve bunlar birçok saldırı türüyle birçok farklı yerden gelebilir. Bir saldırgan, uygulamadaki zafiyetten yararlandıktan sonra kötü amaçlı kod çalıştırabilir, hassas kaynaklara erişebilir, hedef sistemi bir botnet'e dahil edebilir veya kötü amaçlı yazılım ya da fidye yazılımı enjeksiyonları için yollar oluşturabilir.

Genel olarak zafiyetler, donanım, yazılım veya prosedürlerin geliştirilmesi ve uygulanması sırasında yetersiz test, tasarım kusurları, yapılandırma hataları ve insan hatası gibi çeşitli nedenlerden dolayı ortaya çıkar. Web uygulamalarında en çok görülen on zafiyet, son derece kabul gören ve güvenlik ortamı değiştikçe sık sık güncellenen OWASP Top 10 listesinde ayrıntılı olarak açıklanmıştır.

2. OWASP TOP 10

OWASP Top 10, OWASP (The Open Web Application Security Project) tarafından yayımlanan, geliştiriciler ve web uygulaması güvenliği için standart bir farkındalık belgesidir. Web uygulamalarına yönelik en kritik güvenlik riskleri hakkında geniş bir fikir birliğini temsil eder. Böylelikle web geliştiricileri için olası zafiyet açıkları karşısında fikir sahibi olmalarını sağlar.

OWASP Top 10, yıllar içerisinde güvenlik tehditlerinin evrimine ve gelişen teknolojilere paralel olarak değişiklikler göstermiştir. İlk kez 2003 yılında yayınlanan listede, zamanla web uygulamalarındaki risklerin nasıl çeşitlendiği ve karmaşıklaştığı gözlemlenmiştir ve Şekil 2.1.’de gözüktüğü gibi zamanla listede değişikliğe gidilmiştir.



Şekil 2.1. OWASP Top 10 Kategori Değişimi

Listenin 2021'den beri kullanılan güncel versiyonundaki zafiyet kategorileri şu şekildedir:

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery (SSRF)

2.1. Broken Access Control

Broken Access Control, uygulamalarda yapılan yanlış veya yetersiz erişim kontrolü ayarları sonucu, yetkisiz kullanıcıların uygulama içerisinde yer alan kaynaklara erişmesine olanak sağlayan bir güvenlik açığıdır. Bu açık, yetkili olmayan bir kullanıcının sisteme veya uygulamaya erişim hakkı olmadığı halde, bu kaynaklara erişmesine veya bunları değiştirmesine neden olabilir. Yaygın şekilde gözlemlenen Broken Access Control zafiyet sebepleri şu şekildedir:

- Erişim kontrolünün eksik, hatalı sağlanması veya hiç sağlanmaması.
- URL'i veya HTML sayfasını değiştirerek ya da bir saldırı aracı kullanarak API isteklerini değiştirme yoluyla erişim kontrollerini atlanması.

- Benzersiz tanımlayıcıyı sağlayarak (ID gibi) başka birinin hesabının görüntülenmesine veya düzenlenmesine izin verilmesi.
- POST, PUT ve DELETE metotlarıyla eksik erişim kontrolüne sahip API'ye erişilmesi.
- Giriş yapmadan bir kullanıcı gibi davranılması veya kullanıcı olarak giriş yapıldığında bir yönetici gibi davranılması.
- JSON Web Token (JWT) erişim kontrol belirtecini, bir çerezi veya gizli bir alanı manipüle ederek ayrıcalıkların yükseltilmesi ya da JWT'yi geçersiz kılarak meta veri manipülasyonu yapılması.
- Yanlış Cross-Origin Resource Sharing (CORS) yapılandırmasıyla API'ye yetkisiz veya güvenilmeyen kaynaklardan erişilmesi.
- Kimliği doğrulanmamış bir kullanıcı olarak kimliği doğrulanmış sayfalara veya standart bir kullanıcı olarak ayrıcalıklı sayfalara zorla erişilmesi.

Bu zafiyeti önlemek için şu şekilde önlemler alınabilir:

- Erişim kontrolleri düzgün yapılandırmak.
- Token-based authentication kullanmak.
- İşlemlerin logları saklamak.
- Otomatik saldırı araçlarını engellemek için API'lere rate limit eklemek.
- Web sunucusu dizin listelemesini devre dışı bırakmak
- Dosya meta verilerini ve yedekleme dosyalarını kök dizinde saklamamak.

2.2. Cryptographic Failures

Cryptographic Failures, kriptografi kullanılarak korunan verilerin, yanlış bir şekilde şifrelendiği veya şifresinin çözüldüğü durumlardır. Bu tür hatalar, uygulamalarda kullanılan şifreleme algoritmalarının yanlış seçilmesi veya uygulanması, anahtar yönetimi hataları veya rastgele sayı üretimindeki eksiklikler nedeniyle oluşabilir. Bu zafiyet sonucu saldırganlar, korunan verilere erişim sağlayabilir. Cryptographic Failures zafiyet sebepleri şu şekildedir:

- Verilerin şifrelenmeden iletilmesi, HTTP, SMTP, FTP gibi protokollerin kullanılması.
- Verilerin şifrelenmesinde eski veya zayıf şifreleme algoritmaları kullanılması.
- Varsayılan şifreleme anahtarı kullanılması veya zayıf bir anahtar seçilmesi.
- Veri şifreleme zorunluluğunun belirtilmemesi.

- Şifrelemede kullanılan rastgeleliğin öngörülebilir veya düşük entropili olması.

Cryptographic Failures zafiyetinden korunmak için alınılması gereken birkaç önlem:

- Doğru şifreleme algoritmaları kullanılmak.
- Hassas verileri taşımak için FTP ve SMTP gibi eski protokolleri kullanmamak.
- Anahtar yönetimi hatalarından kaçınılmak.
- Saklanmasına gerek olmayan hassas verileri silmek.
- Hassas veriler içeren cevapları önbellekte saklamamak.
- Rastgeleliği doğru sağlamak.
- Şifrelemede kullanılan anahtarları düzenli olarak yenilemek.

2.3. Injection

Injection, web uygulamalarında sıklıkla görülen bir güvenlik açığıdır. Bu açık, uygulamalarda kullanılan veri girişleri yoluyla, kötü niyetli kullanıcıların uygulama tarafından yürütülen veritabanı sorgularına, komutlara veya diğer işlevlere veri girişlerinde yapılan hatalı denetlemeler veya filtrelemeler nedeniyle zararlı kod enjekte etmesini sağlar. En çok kullanılan bazı injection çeşitleri olarak SQL, NoSQL, OS komutu örnek gösterilebilir. Kaynak kodu incelemesi, uygulamaların Injection zafiyetine karşı savunmasız olup olmadığını tespit etmenin en iyi yöntemidir. Injection zafiyet sebepleri şu şekildedir:

- Kullanıcı tarafından sağlanan verilerin, uygulama tarafından doğrulanmaması, filtrelenmemesi veya sterilize edilmemesi.
- Kullanıcı tarafından gelen verilerin doğrudan kullanılması veya birleştirilmesi.

Injection zafiyetinden korunmak için alınabilecek birkaç önlem şu şekildedir:

- SQL parametreleştirme kullanmak.
- Uygulamayı kodlama standartlarına göre geliştirmek.
- WAF kullanmak.
- Kullanıcı tarafından gelen verileri kullanmadan önce doğrulamak, filtrelemek, sterilize etmek.
- Yorumlayıcıyı tamamen kullanmaktan kaçınan, parametreli bir arayüz sağlayan güvenli bir API kullanmak.

2.4. Insecure Design

Insecure Design, bir web uygulamasının tasarımında yapılan hatalar veya eksiklikler nedeniyle ortaya çıkan bir güvenlik açığıdır. Bu, uygulamanın tasarımında yapılan hataların, uygulamanın tüm yaşam döngüsü boyunca devam etmesine ve güvenliğini olumsuz etkilemesine neden olabilir. Güvensiz tasarım, uygulamanın özellikle kimlik doğrulama, yetkilendirme, veri gizliliği ve bütünlüğü gibi önemli güvenlik konularında hatalar içermesiyle ortaya çıkabilir. Uygulama geliştirirken güvenlik odaklı yazılım yaşam döngüsü kullanmamak bu zafiyete sebebiyet verebilir.

Insecure Design zafiyetini önlemek için şu önlemler alınabilir:

- Güvenli tasarım ilkeleri uygulamak.
- Güvenlik odaklı bir yazılım yaşam döngüsü kullanmak.
- Güvenlik açıklarını düzeltmek.
- Uygulama güvenliği için en iyi uygulamalar kullanmak.
- Uygulama geliştirirken AppSec uzmanlarından yardım almak.

2.5. Security Misconfiguration

Security Misconfiguration, bir uygulamanın ya da sistemlerin yanlış yapılandırılması veya yapılandırılmamış olması sonucu oluşan bir güvenlik açığıdır. Bu açık, sistemin, uygulamanın veya sunucuların güvenliğini sağlamak için gereken en iyi uygulamaların uygulanmamış veya eksik uygulanmış olmasından kaynaklanabilir. Bir web uygulaması aşağıdaki durumlarda zafiyete sahip olabilir:

- Bulut hizmetlerinde yanlış yapılandırılmış izinlerin olması.
- Varsayılan hesapların ve şifrelerin hala etkin olması veya değiştirilmemesi.
- Güncellenmiş sistemlerde güvenlik özelliklerinin devre dışı bırakılması ya da yanlış yapılandırılması.
- Kullanılan yazılımın güncel olmaması veya halihazırda zafiyete sahip olması.

Security Misconfiguration zafiyetinden korunmak için alınması gereken önlemler:

- En iyi uygulamaları kullanmak.
- Yazılımın güvenlik düzeyini kontrol etmek.

- Güvenlik yamalarını ve güncellemeleri takip etmek ve uygulamak.
- Sistem yapılandırmasını güncellemek.

2.6. Vulnerable and Outdated Components

Vulnerable and Outdated Components, bir uygulamada kullanılan üçüncü taraf bileşenlerin güncellenmemesi veya bilinen güvenlik açıklarına sahip olması nedeniyle oluşan bir güvenlik açığıdır. Birçok modern uygulama, üçüncü taraf bileşenlerin kullanımını içerir. Bu bileşenler, genellikle web uygulama çerçeveleri, veritabanı yönetim sistemleri, açık kaynak kütüphaneler, sunucu yazılımı ve diğer araçlar gibi yazılım bileşenleri olabilir. Bu zafiyete sebep olan durumlardan birkaçı şu şekildedir:

- Yazılımın desteklenmemesi, güncel olmaması veya halihazırda zafiyete sahip olması.
- Düzenli olarak güvenlik taraması yapılmaması.
- Kullanılan bileşenlerin versiyonlarından bihaber olunması.
- Güncellenmiş, yükseltilmiş veya yamalı kitaplıkların uyumluluğunu test edilmemesi.
- Kullanılan bileşenlerin güvenliğinin yapılandırılmaması.

Bu zafiyeti önlemek için şu şekilde önlemler alınabilir:

- Bileşenleri takip etmek.
- Güncelleme politikaları oluşturmak.
- Kullanılmayan bağımlılıkları, özellikleri, bileşenleri vs kaldırmak.
- Bileşenleri güvenli bağlantılar üzerinden resmi kaynaklardan edinmek.

2.7. Identification and Authentication Failures

Identification and Authentication Failures, bir kullanıcının kimliğinin doğrulanması veya yetkilendirilmesi sırasında yaşanan sorunlar nedeniyle oluşan bir güvenlik açığıdır. Bu tür bir açık, bir saldırganın bir kullanıcının kimliğini çalmasına veya sahte bir kimlik kullanarak uygulamaya erişmesine izin verebilir. Bir web uygulaması aşağıdaki durumlarda bu zafiyete sahip olabilir:

- Brute force veya diğer otomatik saldırıların engellenmemesi.
- Varsayılan, zayıf veya çokça kullanılan kimlik bilgilerinin kullanılması.

- Eksik, zayıf veya etkisiz çok faktörlü kimlik doğrulaması kullanılması.
- Session ID’lerin doğru şekilde geçersiz kılınmaması.
- Şifrelenmiş veya zayıf şifrelenmiş veri depoları kullanılması.

Bu zafiyeti önlemek için şu şekilde önlemler alınabilir:

- Güçlü kimlik doğrulama yöntemleri kullanmak.
- Kimlik doğrulama işlemlerini izlemek.
- Kimlik bilgilerini şifrelemek.
- Düzenli olarak kimlik doğrulama politikalarını kontrol etmek.
- Çok faktörlü kimlik doğrulama yöntemlerini kullanmak.
- Güçlü kimlik bilgileri kullanmak.
- Başarısız oturum açma girişimlerini sınırlamak veya geciktirmek.

2.8. Software and Data Integrity Failures

Software and Data Integrity Failures, bir yazılım veya veri sisteminin beklenmeyen şekilde değiştirilmesi bozulması sonucu meydana gelen bir güvenlik açığıdır. Bu zafiyet, bir saldırganın yazılım veya veri sistemini hedef alarak sistemi istismar etmesine veya manipüle etmesine izin verebilir. Saldırgan verileri çalabilir, verileri bozabilir veya sistemi kontrol etmeye başlayabilir.

Software and Data Integrity Failures, bir yazılım güncellemesi sırasında, yazılımın kötü amaçlı bir saldırgan tarafından değiştirilmesi veya bir saldırganın bir veri depolama ortamına kötü amaçlı yazılım yerleştirmesi gibi birçok farklı şekilde ortaya çıkabilir.

Bu zafiyeti önlemek için şu şekilde önlemler alınabilir:

- Yazılım ve veri yedeklemeleri oluşturmak.
- Kütüphanelerin ve bağımlılıkların güvenilir depolar kullandığını kontrol etmek.
- Dijital imza veya benzer mekanizmalar kullanmak.
- Kod ve yapılandırma değişiklikleri için bir inceleme süreci oluşturmak.
- Yazılım güncellemelerini düzenli olarak yüklemek.
- Güvenli yazılım geliştirme yöntemlerini kullanmak.

2.9. Security Logging and Monitoring Failures

Security Logging and Monitoring Failures, güvenlik olaylarının izlenmesi ve kaydedilmesi işlevlerinin yetersizliği veya hatalı yapılandırılması sonucu ortaya çıkan bir zafiyettir. Bu tür bir zafiyet, kötü amaçlı aktivitelerin tespit edilememesi veya güvenlik olaylarına yanıt verilememesi gibi sonuçlara neden olabilir. Bu zafiyete sebep olan durumlardan birkaçı şu şekildedir:

- Logların yalnızca yerel olarak depolanması.
- Uygun uyarı eşikleri ve müdahale süreçlerinin mevcut veya etkili olmaması.
- Uygulama, aktif saldırıları anlık veya anlığa yakın olarak algılayamaması, iletememesi veya uyarı verememesi.
- Uygulama ve API loglarının şüpheli etkinlik açısından izlenmemesi.
- Oturum açma, başarısız oturum açma ve yüksek değerli işlemler gibi denetlenebilir olayların loga kaydedilmemesi.

Bu zafiyeti önlemek için şu şekilde önlemler alınabilir:

- Güvenlik olaylarının izlenmesi ve kaydedilmesi için uygun araçlar kullanmak.
- Günlük kayıtlarını düzenli olarak incelemek.
- Güvenli uyarı ve alarm sistemleri kullanmak.
- Güvenlik politikalarını ve prosedürlerini düzenli olarak gözden geçirmek.
- Log verilerini doğru şekilde şifrelemek.

2.10. Server-Side Request Forgery (SSRF)

Server-Side Request Forgey (SSRF), 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. Modern web uygulamaları kullanıcılara kullanışlı özellikler sağladığından, bir URL'nin getirilmesi yaygın bir senaryo haline gelmiştir.

SSRF, hedef sunucu için ciddi bir güvenlik tehdidi oluşturur, çünkü saldırgan sunucuya istekler göndererek hassas verileri çalabilir, sunucunun kaynaklarını tüketebilir, sunucunun kontrolünü ele geçirebilir veya sunucunun çalışmasını bozabilir.

Bu zafiyeti önlemek için şu şekilde önlemler alınabilir:

- WAF kullanmak.
- Kullanıcı tarafından sağlanan tüm inputları temizlemek ve doğrulamak.
- Sunucu ayarlarını kontrol etmek.
- HTTP yönlendirmelerini devre dışı bırakmak.
- Kullanıcılara işlenmemiş cevaplar göndermemek.
- URL’ler, portlar için whitelist kullanmak.
- Güvenlik duvarında kabul edilen ve engellenen tüm ağ akışlarını kaydetmek.