OWASP

OWASP Nedir?

Açık Web Uygulaması Güvenliği Projesi veya OWASP, web uygulaması güvenliğine adanmış uluslararası bir organizasyondur. Sundukları materyaller dokümantasyonu, araçları, videoları ve forumları içerir. Belki de en iyi bilinen projesi OWASP Top 10'dur.

1. Broken Access Control

Zafiyet Nedir?

Bir uygulamanın, yetkisiz kullanıcıların erişimini engellememesidir. Yani, bir kullanıcı olması gerekenden daha fazla yetkiye sahip olabilir veya yetkisi olmayan bölgelere erişebilir. Bu durum, verilerin çalınması, sistemlerin ele geçirilmesi gibi ciddi sonuçlara yol açabilir. Örneğin, bir kullanıcı kendi profilini düzenlemek yerine tüm kullanıcıların verilerini değiştirebilir veya sadece yöneticilere açık olan bölümlere girebilir.

Neden Kaynaklanır?

Zayıf Kimlik Doğrulama: Kullanıcıların kimliklerini doğrulamak için yetersiz yöntemlerin kullanılması. Örneğin, basit şifreler veya tek faktörlü kimlik doğrulama kullanılması.

Yetkilerin Yanlış Yönetimi: Kullanıcılara verilen yetkilerin sistemin gereksinimlerine uygun olmaması veya yetkilerin zamanında geri alınmaması. Örneğin, bir çalışan işten ayrıldıktan sonra sistemde hala yetkisi olabilir.

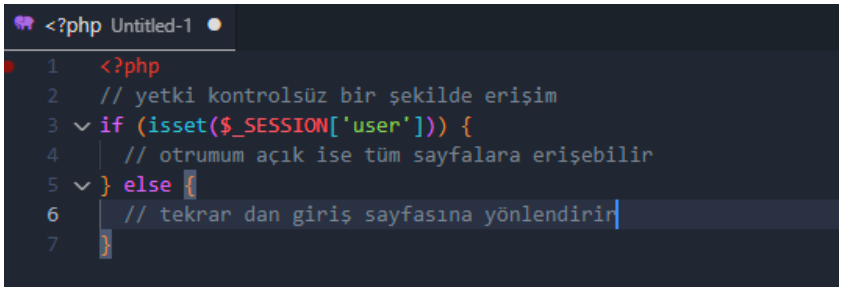
Erişim Kontrol Listelerinin Hatalı Yapılandırılması: Hangi kullanıcının hangi kaynağa erişebileceğini belirleyen listelerin doğru oluşturulmaması. Örneğin, bir herkese açık dosyanın izinleri herkese yazma olarak ayarlanmış olabilir.

Türleri

Yatay Ayrıcalık Yükseltme: Bir kullanıcı, kendi yetki seviyesindeki diğer kullanıcıların verilerine erişebilir. Örneğin, bir satış temsilcisi başka bir satış temsilcisinin müşteri bilgilerini görebilir.

Dikey Ayrıcalık Yükseltme: Bir kullanıcı, daha yüksek seviyedeki bir kullanıcının yetkilerine sahip olabilir. Örneğin, bir normal kullanıcı yönetici paneline erişebilir.

Zafiyetli PHP Kodu



Bu kodda, oturum açmış herhangi bir kullanıcı sitedeki tüm sayfalara erişebilir. Bu durum, yetkisiz erişime açık kapı bırakır.

Nasıl Önlenir?

Güçlü Kimlik Doğrulama Mekanizmaları Kullanın: Şifre karmaşıklığı kuralları, çok faktörlü kimlik doğrulama gibi yöntemlerle kullanıcıların kimliklerini doğrulayın.

Yetkileri En İnce Ayrıntısına Kadar Tanımlayın: Her kullanıcının hangi verilere ve işlemlere erişebileceğini net bir şekilde belirleyin ve bu yetkileri düzenli olarak gözden geçirin.

Erişim Kontrol Listelerini Sürekli Güncelleyin: Sistemde değişiklik olduğunda erişim kontrol listelerini güncellemeyi unutmayın.

Güvenlik Açıklarını Tarayıcı Araçlar Kullanarak Tespit Edin: Düzenli olarak güvenlik taramaları yaparak potansiyel açıkları belirleyin ve düzeltin.

En Küçük İzin Prensibini Uyglayın: Kullanıcılara sadece işlerini yapmaları için gerekli olan izinleri verin.

Oturum Yönetimini Dikkatlice Yapın: Oturum sürelerini sınırlayın, oturum çalıntılarına karşı önlem alın.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve uygulamalarınızı dışarıdan gelebilecek saldırılara karşı koruyun.

2. SQL Injection

Zafiyet Nedir?

SQL Enjeksiyonu, bir web uygulamasının veri tabanına gönderdiği SQL sorgularını manipüle ederek saldırganların izinsiz bir şekilde verilere erişmesini sağlayan bir güvenlik açığıdır. Saldırganlar, kullanıcı giriş alanlarına özel karakterler veya SQL komutları ekleyerek uygulamanın normal işleyişini bozar ve veri tabanından istedikleri bilgilere ulaşabilirler. Bu durum, verilerin çalınması, değiştirilmesi veya silinmesi gibi ciddi sonuçlara yol açabilir.

Neden Kaynaklanır?

Kullanıcı Girişlerinin Doğrudan SQL Sorgularında Kullanılması: Uygulamalar, kullanıcıların girdiği verileri doğrudan SQL sorgularında kullandığında bu durum, SQL enjeksiyonuna açık hale gelir.

Giriş Verilerinin Yetersiz Filtrelenmesi: Giriş verileri üzerinde yeterli temizleme ve doğrulama işlemleri yapılmadığında, saldırganlar özel karakterleri kullanarak SQL sorgularını manipüle edebilir.

Hazırlanmış Sorguların Kullanılmaması: Hazırlanmış sorgular (prepared statements), SQL enjeksiyonuna karşı önemli bir savunma mekanizmasıdır. Ancak birçok uygulama bu mekanizmayı kullanmaz.

Türleri

Sayfa Seçimi: Saldırgan, ziyaret edeceği sayfayı kontrol etmek için SQL sorgusunu manipüle eder.

Veri Değiştirme: Saldırgan, veritabanındaki verileri değiştirmek için SQL sorgusunu manipüle eder.

Veritabanı Yapısını Değiştirme: Saldırgan, veritabanının yapısını değiştirmek için SQL sorgusunu manipüle eder.

Veritabanı Sunucusuna Erişimi Ele Geçirme: Saldırgan, veritabanı sunucusuna komutlar göndererek kontrolü ele geçirebilir.

Zafiyetli PHP Kodu

<?php Untitled-1

```php
<?php
// inputan alınana girdiyi doğrudan sorguya gönderiyor
$username = $_POST['username'];
$password = $_POST['password'];

$sql = "SELECT * FROM users WHERE username='$username' AND password='$password'";
$result = mysqli_query($conn, $sql);
```

Bu kodda, kullanıcının girdiği username değeri doğrudan SQL sorgusunda kullanıldığı için SQL enjeksiyonuna açık bir durum söz konusudur. Örneğin, saldırgan username alanına ' OR '1'='1' gibi bir değer girerek tüm kullanıcıların bilgilerine ulaşabilir.

Nasıl Önlenir?

Hazırlanmış Sorgular Kullanın: Hazırlanmış sorgular, SQL enjeksiyonuna karşı en etkili savunma yöntemidir. Bu yöntemde, SQL sorgusunun yapısı önceden belirlenir ve kullanıcıdan gelen veriler parametre olarak verilir.

Giriş Verilerini Filtreleyin ve Doğrulayın: Kullanıcıdan gelen tüm verileri özel karakterlere karşı kontrol edin ve güvenli hale getirin.

Parametreli Sorguları Kullanın: Veritabanı sürücülerinin sağladığı parametreli sorgu fonksiyonlarını kullanarak SQL enjeksiyonunu önleyin.

Stored Procedure'ları Kullanın: Stored procedure'lar, önceden tanımlanmış SQL komutlarıdır ve SQL enjeksiyonuna karşı daha güvenlidir.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve uygulamalarınızı dışarıdan gelebilecek saldırılara karşı koruyun.

3. Cryptographic Failures

Zafiyet Nedir?

Kriptografik hatalar, bir sistemin verileri şifreleme ve çözme süreçlerinde kullanılan kriptografik algoritmaların doğru ve güvenli bir şekilde uygulanmamasından kaynaklanan güvenlik açıklarıdır. Bu açıklar, yetkisiz erişim, veri sızması ve sistem bütünlüğünün bozulması gibi ciddi sonuçlara yol açabilir.

Neden Kaynaklanır?

Zayıf Algoritmaların Kullanılması: Güvenliği kanıtlanmamış veya kırılmış algoritmaların tercih edilmesi.

Kısa Anahtar Uzunlukları: Kriptografik algoritmaların etkinliği, kullanılan anahtarın uzunluğuyla doğrudan ilişkilidir. Kısa anahtarlar, brute-force saldırılarına karşı daha savunmasızdır.

Zayıf Rastgele Sayı Üretimi: Kriptografik işlemlerde kullanılan rastgele sayılar, tahmin edilebilirse sistemin güvenliği tehlikeye girer.

Yanlış Kriptografik Modların Kullanılması: Şifreleme modlarının yanlış seçimi veya uygulanması, güvenlik açıklarına neden olabilir.

Kriptografik Anahtarların Güvenli Saklanmaması: Anahtarların düz metin olarak saklanması veya zayıf şifreleme yöntemleriyle korunması, anahtarların çalınma riskini artırır.

Türleri

Zayıf Şifreleme Algoritmaları: DES, MD5 gibi artık güvenli kabul edilmeyen algoritmaların kullanılması.

Kısa Anahtar Uzunlukları: 128 bitten daha kısa anahtarların kullanılması.

Zayıf Rastgele Sayı Üretimi: Tahmin edilebilir rastgele sayılar kullanılması.

Yanlış Kriptografik Modların Kullanılması: ECB (Electronic Codebook) gibi güvenliği zayıf modların kullanılması.

Kriptografik Anahtar Sızıntıları: Anahtarların yanlışlıkla ifşa edilmesi veya çalınması.

Örnek

Zayıf Şifreleme Algoritması Örneği: Bir web uygulamasında tüm trafiği şifrelemek için DES algoritması kullanılması. DES algoritması, günümüzde kırılmış kabul edildiği için bu durum ciddi bir güvenlik açığıdır.

Kısa Anahtar Uzunluğu Örneği: Bir web sitesi için 64 bitlik bir şifreleme anahtarı kullanılması. Bu anahtar, modern bilgisayarlar tarafından kısa sürede kırılabilir.

Nasıl Önlenir?

Güçlü Algoritmalar Kullanın: AES, SHA-256 gibi güncel ve güvenli kabul edilen algoritmaları tercih edin.

Yeterli Anahtar Uzunlukları Kullanın: En az 128 bitlik anahtar uzunlukları kullanın.

Güçlü Rastgele Sayı Üreticileri Kullanın: Kriptografik olarak güvenli rastgele sayı üreticileri kullanın.

Doğru Kriptografik Modları Kullanın: CBC, GCM gibi güvenli modları tercih edin.

Kriptografik Anahtarları Güvenli Bir Şekilde Saklayın: Anahtarları şifreleyerek veya güvenli donanım modüllerinde saklayarak koruyun.

Düzenli Güncellemeler Yapın: Kütüphaneleri ve yazılımları düzenli olarak güncelleyerek bilinen güvenlik açıklarını kapatın.

4. Insecure Design

Zafiyet Nedir?

Insecure Design (Güvensiz Tasarım), bir uygulamanın başlangıcından itibaren güvenlik açığı içerecek şekilde tasarlanması veya geliştirilmesidir. Bu tür zafiyetler, uygulamanın tüm yaşam döngüsü boyunca risk oluşturur ve düzeltmesi zor olabilir.

Neden Kaynaklanır?

- **Güvenlik Bilgisinin Eksikliği:** Uygulama geliştiricilerinin güvenlik konusunda yeterli bilgi ve deneyime sahip olmaması.
- **Hızlı Geliştirme Baskısı:** Uygulamanın hızlı bir şekilde piyasaya sürülmesi gerektiği için güvenlik önlemlerinin ihmal edilmesi.
- **Eski Teknolojilerin Kullanımı:** Güncel güvenlik standartlarını karşılamayan eski teknolojilerin kullanılması.
- **Karmaşık Tasarımlar:** Uygulamanın karmaşık ve anlaşılması zor bir yapıya sahip olması, güvenlik açıklarının tespitini zorlaştırır.

Türleri

- **Fonksiyonel Güvenlik Açıkları:** Uygulamanın temel işlevlerinin güvenlik açıkları içermesi.
- **Mimari Güvenlik Açıkları:** Uygulamanın genel mimarisinin güvenlik riskleri taşıması.
- **Veri Güvenliği Açıkları:** Verilerin güvenli bir şekilde işlenmesi ve saklanması konusunda eksiklikler olması.
- **Kimlik Doğrulama ve Yetkilendirme Güvenlik Açıkları:** Kullanıcıların kimliklerinin doğrulama ve yetkilerinin yönetimi konusunda hatalar olması.

Örnekler

- **Fonksiyonel Güvenlik Açığı Örneği:** Bir e-ticaret uygulamasının ödeme bilgilerini düz metin olarak göndermesi.
- **Mimari Güvenlik Açığı Örneği:** Bir web uygulamasının tüm iş mantığının sunucu tarafında gerçekleştirilmesi, saldırganların sunucuya doğrudan erişim elde etmesi riskini artırır.
- **Veri Güvenliği Açığı Örneği:** Bir uygulama, kullanıcıların şifrelerini düz metin olarak veri tabanında saklaması.
- **Kimlik Doğrulama ve Yetkilendirme Güvenlik Açığı Örneği:** Bir uygulama, kullanıcıların kimliklerini doğrulamak için zayıf şifreleme algoritmaları kullanması veya yetkilerin doğru bir şekilde yönetilmemesi.

Nasıl Önlenir?

- **Güvenlik Bilgisini Artırın:** Uygulama geliştiricilerinin güvenlik konusunda eğitim almasını sağlayın.
- **Güvenlik Odaklı Geliştirme Süreçleri Kullanın:** Güvenlik testlerini geliştirme sürecinin erken aşamalarına dahil edin.

- **Güncel Teknolojiler Kullanın:** Güvenlik açığı bildirimleri ve düzeltmelerini takip edin.
- **Basit ve Anlaşılır Tasarımlar Tercih Edin:** Uygulamanın karmaşıklığını minimize edin.
- **Güvenlik İncelemeleri Yapın:** Uygulamanın güvenliğini düzenli olarak gözden geçirin.
- **Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın:** Ağınızı ve uygulamalarınızı dışarıdan gelebilecek saldırılara karşı koruyun.

5. Security Misconfiguration

Zafiyet Nedir?

Security Misconfiguration (Güvenlik Yanlış Yapılandırması), bir sistemin güvenlik ayarlarının doğru şekilde yapılandırılmaması veya varsayılan ayarların kullanılması nedeniyle ortaya çıkan güvenlik açıklarıdır. Bu açıklar, yetkisiz erişim, veri sızması ve sistemin ele geçirilmesi gibi ciddi sonuçlara yol açabilir.

Neden Kaynaklanır?

Varsayılan Ayarların Kullanılması: Sistemlerin veya yazılımların varsayılan ayarları genellikle güvenlik açısından riskli olabilir.

Yanlış Yapılandırma: Güvenlik ayarlarının doğru şekilde yapılandırılmaması veya eksik konfigürasyonlar.

Güncellemelerin Uygulamaması: Sistemlerin ve yazılımların güvenlik güncellemelerinin zamanında uygulanmaması.

Yetkisiz Erişimlerin Ortaya Çıkması: Yetkisiz kullanıcıların sistemlere erişim elde etmesi.

Türleri

Varsayılan Ayarlar: Sistemlerin veya yazılımların varsayılan kullanıcı adları, şifreler veya diğer ayarların kullanılması.

Yetkisiz Erişimler: Yetkisiz kullanıcıların sistemlere erişim elde etmesi.

Güvenlik Güncellemelerinin Uygulamaması: Sistemlerin ve yazılımların güvenlik güncellemelerinin zamanında uygulanmaması.

Yanlış Yapılandırma: Güvenlik duvarları, firewall kuralları, erişim kontrol listeleri gibi güvenlik ayarlarının yanlış yapılandırılması.

Örnekler

Varsayılan Ayarlar Örneği: Bir web sunucusunun varsayılan kullanıcı adı ve şifresiyle kullanılması.

Yetkisiz Erişimler Örneği: Bir sistemin, yetkisiz kullanıcıların erişimine açık bırakılması.

Güvenlik Güncellemelerinin Uygulamaması Örneği: Bir sistemin, güvenlik açıklarını gideren güncellemelerin uygulanmaması.

Yanlış Yapılandırma Örneği: Bir güvenlik duvarının, tüm trafiğe izin verecek şekilde yanlış yapılandırılması.

Nasıl Önlenir?

Varsayılan Ayarları Değiştirin: Sistemlerin ve yazılımların varsayılan ayarlarını güvenli değerlerle değiştirin.

Yetkisiz Erişimleri Engelleyin: Sistemlere erişimi yetkili kişilerle sınırlayın.

Güvenlik Güncellemelerini Uygulayın: Sistemlerin ve yazılımların güvenlik güncellemelerini zamanında uygulayın.

Güvenlik Ayarlarını Doğru Yapılandırın: Güvenlik duvarları, firewall kuralları, erişim kontrol listeleri gibi güvenlik ayarlarını doğru şekilde yapılandırın.

Güvenlik İncelemeleri Yapın: Sistemlerin güvenlik durumunu düzenli olarak gözden geçirin.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve sistemlerinizi dışarıdan gelebilecek saldırılara karşı koruyun.

6. Vulnerable and Outdated Components

Zafiyet Nedir?

Vulnerable and Outdated Components (Savunmasız ve Eski Bileşenler), bir sistemin kullandığı üçüncü taraf yazılımlar, bileşenler veya kütüphanelerin güvenlik açıklarına sahip olması veya güncellemelerin uygulanmaması nedeniyle ortaya çıkan güvenlik riskleridir. Bu riskler, yetkisiz erişim, veri sızması ve sistemin ele geçirilmesi gibi ciddi sonuçlara yol açabilir.

Neden Kaynaklanır?

Eski Bileşenlerin Kullanımı: Sistemlerin eski ve desteklenmeyen bileşenleri kullanması.

Güvenlik Açıklarının Ortaya Çıkması: Üçüncü taraf yazılımlar, bileşenler veya kütüphanelerde güvenlik açıklarının keşfedilmesi.

Güncellemelerin Uygulamaması: Sistemlerin ve bileşenlerinin güvenlik güncellemelerinin zamanında uygulanmaması.

Bağımlılık Yönetiminin Zayıflığı: Sistemlerin üçüncü taraf bileşenlerine olan bağımlılıklarının doğru şekilde yönetilmemesi.

Türleri

Eski Bileşenler: Sistemlerin eski ve desteklenmeyen yazılımlar, bileşenler veya kütüphaneleri kullanması.

Güvenlik Açıkları: Üçüncü taraf yazılımlar, bileşenler veya kütüphanelerde keşfedilen güvenlik açıkları.

Güncellemelerin Uygulamaması: Sistemlerin ve bileşenlerinin güvenlik güncellemelerinin zamanında uygulanmaması.

Bağımlılık Yönetimi Hataları: Bağımlılıkların çakışması, güncellemelerin uygulanmaması gibi sorunlar.

Örnekler

Eski Bileşenler Örneği: Bir sistemin, desteklenmeyen bir web sunucu yazılımı kullanması.

Güvenlik Açıkları Örneği: Bir sistemin kullandığı bir kütüphanede, uzaktan kod yürütme açıklığı bulunması.

Güncellemelerin Uygulamaması Örneği: Bir sistemin, güvenlik açıklarını gideren güncellemelerin uygulanmaması.

Bağımlılık Yönetimi Hataları Örneği: Bir sistemde, farklı bileşenlerin aynı kütüphanenin farklı sürümlerine bağımlı olması.

Nasıl Önlenir?

Güncel Bileşenler Kullanın: Sistemlerin güncel ve desteklenen bileşenleri kullanmasını sağlayın.

Güvenlik Açıklarını İzleyin: Üçüncü taraf yazılımlar, bileşenler ve kütüphanelerdeki güvenlik açıklarını izleyin.

Güncellemeleri Uygulayın: Sistemlerin ve bileşenlerinin güvenlik güncellemelerini zamanında uygulayın.

Bağımlılık Yönetimini Geliştirin: Bağımlılıkların doğru şekilde yönetilmesini sağlayın.

Güvenlik İncelemeleri Yapın: Sistemlerin güvenlik durumunu düzenli olarak gözden geçirin.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve sistemlerinizi dışarıdan gelebilecek saldırılara karşı koruyun.

7. Identification and Authentication Failures

Zafiyet Nedir?

Identification and Authentication Failures (Kimlik Doğrulama ve Yetkilendirme Hataları), bir sistemin kullanıcıların kimliklerini doğrulama ve yetkilerini doğru bir şekilde yönetme konusunda güvenlik açıklarına sahip olmasıdır. Bu açıklar, yetkisiz erişim, veri sızması ve hesap ele geçirme gibi ciddi sonuçlara yol açabilir.

Neden Kaynaklanır?

Zayıf Kimlik Doğrulama Mekanizmaları: Kullanıcıların kimliklerini doğrulamak için yetersiz yöntemlerin kullanılması.

Yetkisiz Erişimlerin Ortaya Çıkması: Yetkisiz kullanıcıların sistemlere erişim elde etmesi.

Güvenlik Güncellemelerinin Uygulamaması: Sistemlerin ve bileşenlerinin güvenlik güncellemelerinin zamanında uygulanmaması.

Bağımlılık Yönetimi Hataları: Bağımlılıkların çakışması, güncellemelerin uygulanmaması gibi sorunlar.

Türleri

Zayıf Kimlik Doğrulama: Kullanıcıların kimliklerini doğrulamak için yetersiz yöntemlerin kullanılması.

Yetkisiz Erişimler: Yetkisiz kullanıcıların sistemlere erişim elde etmesi.

Güvenlik Güncellemelerinin Uygulamaması: Sistemlerin ve bileşenlerinin güvenlik güncellemelerinin zamanında uygulanmaması.

Bağımlılık Yönetimi Hataları: Bağımlılıkların çakışması, güncellemelerin uygulanmaması gibi sorunlar.

Örnekler

Zayıf Kimlik Doğrulama Örneği: Bir sistemin, sadece kullanıcı adı ve şifre ile kimlik doğrulaması yapması.

Yetkisiz Erişimler Örneği: Bir sistemin, yetkisiz kullanıcıların erişimine açık olması.

Güvenlik Güncellemelerinin Uygulamaması Örneği: Bir sistemin, kimlik doğrulama mekanizmalarını güncellememesi.

Bağımlılık Yönetimi Hataları Örneği: Bir sistemin, kimlik doğrulama mekanizmalarının bağımlılıklarının doğru şekilde yönetilmemesi.

Nasıl Önlenir?

Güçlü Kimlik Doğrulama Mekanizmaları Kullanın: Kullanıcıların kimliklerini doğrulamak için güçlü yöntemler kullanın, örneğin çok faktörlü kimlik doğrulama.

Yetkisiz Erişimleri Engelleyin: Sistemlere erişimi yetkili kişilerle sınırlayın.

Güvenlik Güncellemelerini Uygulayın: Kimlik doğrulama mekanizmalarının güvenlik güncellemelerini zamanında uygulayın.

Bağımlılık Yönetimini Geliştirin: Kimlik doğrulama mekanizmalarının bağımlılıklarının doğru şekilde yönetilmesini sağlayın.

Güvenlik İncelemeleri Yapın: Sistemlerin güvenlik durumunu düzenli olarak gözden geçirin.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve sistemlerinizi dışarıdan gelebilecek saldırılara karşı koruyun.

8. Software and Data Integrity Failures

Zafiyet Nedir?

Software and Data Integrity Failures (Yazılım ve Veri Bütünlüğü Hataları), bir sistemin yazılım ve veri bütünlüğünü bozan güvenlik açıklarıdır. Bu açıklar, veri sızması, sistemin işlevselliğinin bozulması ve güvenlik risklerinin artması gibi sonuçlara yol açabilir.

Neden Kaynaklanır?

Zayıf Veri Doğrulama: Verilerin doğrulama ve temizleme işlemlerinin eksik veya hatalı olması.

Veri Sızıntıları: Verilerin yetkisiz kişiler tarafından erişilebilir hale gelmesi.

Yazılım Hataları: Yazılım kodlarındaki hataların güvenlik açıklarına yol açması.

Güncellemelerin Uygulamaması: Yazılım ve bileşenlerin güvenlik güncellemelerinin zamanında uygulanmaması.

Türleri

Veri Doğrulama Hataları: Verilerin doğrulama ve temizleme işlemlerinin eksik veya hatalı olması.

Veri Sızıntıları: Verilerin yetkisiz kişiler tarafından erişilebilir hale gelmesi.

Yazılım Hataları: Yazılım kodlarındaki hataların güvenlik açıklarına yol açması.

Güncellemelerin Uygulamaması: Yazılım ve bileşenlerin güvenlik güncellemelerinin zamanında uygulanmaması.

Örnekler

Veri Doğrulama Hataları Örneği: Bir sistemin, kullanıcı tarafından girilen verileri doğrulamadan doğrudan kullanması.

Veri Sızıntıları Örneği: Bir sistemin, verileri şifrelenmemiş bir şekilde ağ üzerinden göndermesi.

Yazılım Hataları Örneği: Bir sistemin, bellek taşması veya enjeksiyon saldırılarına açık olması.

Güncellemelerin Uygulamaması Örneği: Bir sistemin, güvenlik açıklarını gideren güncellemelerin uygulanmaması.

Nasıl Önlenir?

Veri Doğrulama Mekanizmaları Kullanın: Verilerin doğrulama ve temizleme işlemlerini gerçekleştirin.

Veri Sızıntılarını Önleyin: Verileri şifreleyin ve güvenli bir şekilde saklayın.

Yazılım Kalitesine Önem Verin: Yazılım geliştirme süreçlerinde kalite kontrol mekanizmaları kullanın.

Güvenlik Güncellemelerini Uygulayın: Yazılım ve bileşenlerin güvenlik güncellemelerini zamanında uygulayın.

Güvenlik İncelemeleri Yapın: Sistemlerin güvenlik durumunu düzenli olarak gözden geçirin.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve sistemlerinizi dışarıdan gelebilecek saldırılara karşı koruyun.

9. Security Logging and Monitoring Failures

Zafiyet Nedir?

Security Logging and Monitoring Failures (Güvenlik Günlüğü ve İzleme Hataları), bir sistemin güvenlik olaylarını yeterli şekilde günlüğe kaydetmemesi veya izlememesi nedeniyle ortaya çıkan güvenlik riskleridir. Bu riskler, güvenlik olaylarının tespiti, analizi ve önlenmesini zorlaştırır.

Neden Kaynaklanır?

Eksik veya Yanlış Günlük Kaydı: Sistemlerin güvenlik olaylarını yeterli şekilde günlüğe kaydetmemesi veya günlük kayıtlarının yanlış yapılandırılması.

Günlüklerin İncelenmemesi: Günlüklerin düzenli olarak incelenmemesi veya analiz edilmemesi.

İzleme Sistemlerinin Eksikliği: Güvenlik olaylarını izlemek için yeterli araçların bulunmaması veya kullanılmaması.

İzleme Sistemlerinin Yanlış Yapılandırması: İzleme sistemlerinin doğru şekilde yapılandırılmaması.

Türleri

Eksik veya Yanlış Günlük Kaydı: Sistemlerin güvenlik olaylarını yeterli şekilde günlüğe kaydetmemesi veya günlük kayıtlarının yanlış yapılandırılması.

Günlüklerin İncelenmemesi: Günlüklerin düzenli olarak incelenmemesi veya analiz edilmemesi.

İzleme Sistemlerinin Eksikliği: Güvenlik olaylarını izlemek için yeterli araçların bulunmaması veya kullanılmaması.

İzleme Sistemlerinin Yanlış Yapılandırması: İzleme sistemlerinin doğru şekilde yapılandırılmaması.

Örnekler

Eksik veya Yanlış Günlük Kaydı Örneği: Bir sistemin, sadece başarılı girişleri günlüğe kaydetmesi.

Günlüklerin İncelenmemesi Örneği: Günlüklerin aylarca incelenmemesi.

İzleme Sistemlerinin Eksikliği Örneği: Bir sistemin, güvenlik olaylarını izlemek için herhangi bir araç kullanmaması.

İzleme Sistemlerinin Yanlış Yapılandırması Örneği: Bir izleme sisteminin, kritik güvenlik olaylarını filtrelemesi.

Nasıl Önlenir?

Etkili Günlük Kaydı Sistemleri Kullanın: Sistemlerin güvenlik olaylarını ayrıntılı bir şekilde günlüğe kaydetmesini sağlayın.

Günlükleri Düzenli Olarak İnceleyin: Günlükleri düzenli olarak analiz edin ve anormal aktiviteleri tespit edin.

İzleme Sistemleri Kullanın: Güvenlik olaylarını izlemek için uygun araçlar kullanın.

İzleme Sistemlerini Doğru Yapılandırın: İzleme sistemlerini doğru şekilde yapılandırın ve güncelleyin.

Güvenlik İncelemeleri Yapın: Sistemlerin güvenlik durumunu düzenli olarak gözden geçirin.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve sistemlerinizi dışarıdan gelebilecek saldırılara karşı koruyun.

10. Server-Side Request Forgery

Zafiyet Nedir?

Server-Side Request Forgery (Sunucu Tarafı İstek Sahteciliği), bir saldırganın bir web uygulaması aracılığıyla sunucunun başka bir kaynağa isteği yapmasını sağladığı bir güvenlik açığıdır. Bu açık, yetkisiz erişim, veri sızması ve sistemin ele geçirilmesi gibi ciddi sonuçlara yol açabilir.

Neden Kaynaklanır?

Zayıf İstek Doğrulama: Uygulamanın, kullanıcı tarafından sağlanan istekleri doğrulamaması veya yetersiz doğrulaması.

Güvenlik Açıklarına Sahip Bileşenlerin Kullanımı: Uygulamanın, güvenlik açıklarına sahip üçüncü taraf bileşenleri kullanması.

Yanlış Yapılandırma: Uygulamanın güvenlik ayarlarının yanlış yapılandırılması.

Türleri

İstek Enjeksiyonu: Saldırganın, uygulamanın isteği oluşturma mekanizmasını kullanarak isteği değiştirerek başka bir kaynağa yönlendirmesi.

İstek Yönlendirme: Saldırganın, uygulamanın isteği yönlendirme mekanizmasını kullanarak isteği başka bir kaynağa yönlendirmesi.

İstek Sahteciliği: Saldırganın, uygulamanın isteği oluşturma mekanizmasını kullanarak sahte bir istek oluşturması.

Örnekler

İstek Enjeksiyonu Örneği: Bir uygulama, kullanıcının sağladığı bir URL'yi doğrudan bir HTTP isteği olarak gönderirse, saldırgan bu URL'yi değiştirerek başka bir kaynağa yönlendirebilir.

İstek Yönlendirme Örneği: Bir uygulama, kullanıcının sağladığı bir URL'yi başka bir kaynağa yönlendirmek için kullanıyorsa, saldırgan bu URL'yi değiştirerek isteği başka bir kaynağa yönlendirebilir.

İstek Sahteciliği Örneği: Bir uygulama, kullanıcının sağladığı bir URL'yi doğrudan bir HTTP isteği olarak gönderirse, saldırgan bu URL'yi değiştirerek sahte bir istek oluşturabilir.

Nasıl Önlenir?

İstek Doğrulama Mekanizmaları Kullanın: Uygulamanın, kullanıcı tarafından sağlanan istekleri doğrulamasını sağlayın.

Güvenli Bileşenler Kullanın: Uygulamanın, güvenli ve güncel üçüncü taraf bileşenleri kullanmasını sağlayın.

Güvenlik Ayarlarını Doğru Yapılandırın: Uygulamanın güvenlik ayarlarını doğru şekilde yapılandırın.

Güvenlik İncelemeleri Yapın: Uygulamanın güvenlik durumunu düzenli olarak gözden geçirin.

Güvenlik Duvarları ve Saldırı Önleme Sistemleri Kullanın: Ağınızı ve uygulamalarınızı dışarıdan gelebilecek saldırılara karşı koruyun.