OWASP TOP 10

1.Broken Access Control

Broken Access Control, bir web uygulamasının kullanıcıların hangi kaynaklara erişebileceğini düzgün bir şekilde kontrol edememesi durumunu ifade eder. Bu tür zafiyetler, yetkisiz kullanıcıların veya düşük yetkilere sahip kullanıcıların, aslında erişimlerinin olmaması gereken verileri görüntülemelerine, düzenlemelerine veya silmelerine olanak tanır.

Nedenleri:

Yanlış veya Eksik Yetkilendirme: Kullanıcıların rollerine göre doğru erişim seviyelerinin belirlenmemesi.

Eksik Otomatik Testler: Erişim kontrolünün düzenli ve kapsamlı bir şekilde test edilmemesi.

Manuel Hatalar: Geliştiricilerin veya sistem yöneticilerinin yapılandırma hataları.

Nasıl Önlenir:

Rol Tabanlı Erişim Kontrolü (RBAC): Kullanıcıların rollerine ve izinlerine göre erişim kontrollerinin uygulanması.

Düzgün Yetkilendirme Denetimleri: Tüm erişim taleplerinin yetkilendirme kontrollerine tabi tutulması.

URL ve Menü Gizleme: Yetkisiz kullanıcıların görmemesi gereken sayfaların URL'lerinin ve menü seçeneklerinin gizlenmesi.

Düzenli Güvenlik Testleri: Erişim kontrolü zafiyetlerini tespit etmek için otomatik ve manuel güvenlik testlerinin düzenli olarak yapılması.

Güvenli Kodlama Pratikleri: Geliştiricilerin erişim kontrol mekanizmalarını doğru ve güvenli bir şekilde kodlaması.

2.Cryptographic Failures

Cryptographic Failures (Kriptografik Hatalar), güvenli bilgi alışverişini ve veri korumasını sağlamak için kullanılan kriptografik tekniklerin yanlış uygulanması, zayıf şifreleme algoritmalarının kullanılması veya kriptografik anahtarların güvensiz yönetilmesi gibi durumlardan kaynaklanan güvenlik

açıklarıdır. Bu tür hatalar, hassas bilgilerin yetkisiz erişime açılmasına, veri bütünlüğünün bozulmasına veya verilerin çalınmasına yol açabilir.

Nedenleri:

Zayıf veya Eski Şifreleme Algoritmalarının Kullanımı: MD5, SHA-1 gibi artık güvenli kabul edilmeyen algoritmaların kullanımı.

Yanlış Şifreleme Modlarının Kullanımı: Şifreleme modlarının hatalı veya yanlış yapılandırılması.

Anahtar Yönetimindeki Hatalar: Anahtarların yanlış saklanması, güvenli olmayan ortamlarda tutulması veya yanlış kişilere verilmesi.

Sertifika Yönetimi Sorunları: SSL/TLS sertifikalarının yanlış yapılandırılması veya sahte sertifikaların kullanımı.

Şifreleme Anahtarlarının Zayıflığı: Yetersiz uzunlukta anahtarların kullanımı veya aynı anahtarın tekrar kullanılması.

Rastgele Sayı Üreteçlerinin Zayıflığı: Güvensiz veya tahmin edilebilir rastgele sayı üreteçlerinin kullanımı.

Doğrulama ve İmza İşlemlerinde Hatalar: Dijital imzaların veya mesaj doğrulama kodlarının yanlış uygulanması.

Nasıl önlenir:

Güncel ve Güçlü Şifreleme Algoritmalarının Kullanımı: AES-256 gibi güçlü ve güncel şifreleme algoritmaları kullanılmalı.

Doğru Şifreleme Modlarının Kullanımı: Örneğin, CBC mod yerine GCM gibi güvenli modlar tercih edilmeli.

Anahtar Yönetimi Uygulamalarının Geliştirilmesi: Anahtarların güvenli şekilde saklanması, düzenli olarak değiştirilmesi ve sadece yetkili kişiler tarafından erişilebilir olması sağlanmalı.

Sertifikaların Doğru Yönetimi: SSL/TLS sertifikalarının düzenli olarak yenilenmesi ve sertifika otoritelerinin güvenilirliği kontrol edilmeli.

Güçlü Rastgele Sayı Üreteçleri Kullanılması: Kriptografik olarak güvenli rastgele sayı üreteçleri kullanılmalı.

Şifreleme Protokollerinin Güncel Tutulması: Protokoller ve kütüphaneler düzenli olarak güncellenmeli ve bilinen güvenlik açıklarına karşı korunmalı.

Eğitim ve Farkındalık: Geliştiricilerin ve sistem yöneticilerinin kriptografi hakkında bilinçlendirilmesi ve bu alanda eğitim alması sağlanmalı.

3. Injection

Injection (Enjeksiyon), bir saldırganın kötü amaçlı komutları bir uygulamanın giriş verileri üzerinden sisteme enjekte etmesine olanak tanıyan bir güvenlik açığı türüdür. Enjeksiyon saldırıları, uygulamanın veri işlemlerini manipüle ederek, yetkisiz veri erişimi, veri bütünlüğünün bozulması, kimlik doğrulama atlatma ve sistemin tamamen kontrol edilmesi gibi sonuçlar doğurabilir. Bu saldırı türü, SQL Injection, Command Injection, XML Injection gibi çeşitli biçimlerde görülebilir.

Nedenleri:

Kullanıcı Girdilerinin Yetersiz veya Hiç Doğrulanmaması: Kullanıcıdan alınan verilerin, uygulamanın işlem yaptığı yerlere doğrudan aktarılması ve bu verilerin doğrulanmaması.

Zayıf veya Hatalı Kodlama Uygulamaları: Geliştiricilerin güvenlik konusuna yeterince dikkat etmemesi ve kodlama esnasında güvenlik önlemlerini göz ardı etmesi.

Eski veya Güvensiz Kütüphanelerin Kullanımı: Bilinen güvenlik açıkları içeren eski veya güncel olmayan kütüphanelerin kullanımı.

Yanlış Yapılandırılmış Veritabanı Erişimleri: Veritabanı kullanıcılarının fazla yetkilere sahip olması veya güvenlik ayarlarının doğru yapılandırılmaması.

Dinamik Sorguların Kullanımı: SQL sorgularında ve diğer komutlarda kullanıcı girdisinin doğrudan kullanılması ve bu girdilerin temizlenmemesi.

Nasıl önlenir:

Parametrik Sorgular ve Hazır Beyanlar Kullanımı: SQL sorguları gibi komutları oluştururken, kullanıcı girdileri doğrudan sorguya dahil edilmemeli. Parametrik sorgular veya hazırlanan ifadeler (prepared statements) kullanılmalı, bu sayede kullanıcı girdileri veri olarak algılanır, komut olarak değil.

Girdi Doğrulama ve Temizleme: Kullanıcıdan alınan tüm veriler doğrulanmalı ve uygun biçimde temizlenmeli. Örneğin, sadece sayılar içermesi gereken bir alan alfasayısal karakterler içermemeli.

Kapsamlı Hata Yönetimi: Uygulama hataları kullanıcıya detaylı bilgi vermemeli. Hata mesajları genellikle saldırganlara sistem hakkında bilgi sağlar, bu nedenle genel hata mesajları kullanılmalı.

Minimum Yetki İlkesinin Uygulanması: Veritabanı kullanıcılarına sadece gerekli minimum yetkiler verilerek, bir saldırı durumunda sistemin zarar görme olasılığı azaltılmalı.

Dinamik Sorgulardan Kaçınma: Dinamik SQL veya komut satırı oluşturmak yerine, parametrik ve hazır ifadeler kullanılmalı.

Web Uygulama Güvenlik Duvarı (WAF) Kullanımı: Web uygulama güvenlik duvarları, enjeksiyon saldırılarını tespit edebilir ve engelleyebilir.

Kütüphane ve Çerçeveleri Güncel Tutma: Güvenlik açıkları kapatılan güncellemeler düzenli olarak yüklenmeli ve kullanılan kütüphaneler güncel tutulmalı.

4.Insecure Design

Insecure Design (Güvensiz Tasarım), bir yazılımın veya sistemin tasarım aşamasında güvenlik açısından yetersiz planlanması veya uygulanması sonucunda ortaya çıkan güvenlik açıklarını ifade eder. Bu, yazılımın yapısal olarak güvenli olmamasına yol açar ve siber saldırılar gibi tehditlere karşı savunmasız hale getirir. Güvensiz tasarım, genellikle yazılım geliştirme yaşam döngüsünün başlarında yapılan hatalardan kaynaklanır ve sistemin genel güvenlik mimarisini etkileyebilir.

Nedenleri:

Güvenlik Gereksinimlerinin Yetersiz Belirlenmesi: Proje başlangıcında güvenlik gereksinimlerinin yeterince tanımlanmaması ve dikkate alınmaması, güvensiz tasarıma yol açabilir.

Tecrübe ve Bilgi Eksikliği: Güvenli yazılım tasarımı konusunda yeterli bilgiye veya deneyime sahip olmayan ekipler, güvenlik açıklarına neden olabilecek kararlar alabilir.

Güvenlik Odaklı Tasarım Prensiplerinin Göz Ardı Edilmesi: Güvenli yazılım geliştirme için kritik olan güvenlik prensiplerinin (örneğin, minimum yetki ilkesi, güvenli varsayılan yapılandırmalar) uygulanmaması.

Fonksiyonellik Önceliği: Projenin hızla tamamlanmasına veya işlevselliğin artırılmasına odaklanılırken güvenlik konularının geri planda bırakılması.

Yetersiz Tehdit Modelleme ve Risk Analizi: Proje boyunca potansiyel tehditlerin ve güvenlik risklerinin yeterince analiz edilmemesi.

Güvenlik Testlerinin Eksikliği: Güvenlik testlerinin (örneğin, penetrasyon testleri, güvenlik taramaları) tasarım aşamasında ve sonrasında yeterince yapılmaması.

Nasıl önlenir:

Güvenlik Gereksinimlerinin Erken Belirlenmesi: Proje başlangıcında güvenlik gereksinimlerini açıkça tanımlamak ve bu gereksinimleri tasarım sürecine entegre etmek.

Tehdit Modelleme ve Risk Analizi: Projenin erken aşamalarında tehdit modelleme ve risk analizi yapmak, potansiyel güvenlik açıklarını önceden belirlemeye yardımcı olur.

Güvenlik Odaklı Tasarım Prensiplerinin Uygulanması:

- **Minimum Yetki İlkesi: Kullanıcılara ve sistem bileşenlerine sadece gerekli yetkilerin verilmesi.**
- **Güvenli Varsayılan Yapılandırmalar: Sistemlerin varsayılan ayarlarının güvenli olacak şekilde yapılandırılması.**
- **Savunma Derinliği (Defense in Depth): Birden fazla güvenlik katmanı kullanarak güvenlik açıklarını azaltmak.**

Güvenli Kodlama Standartları Kullanma: Güvenli kodlama standartları ve en iyi uygulamalar (best practices) belirleyerek, ekip üyelerinin bu standartlara uygun olarak kod yazmasını sağlamak.

Düzenli Güvenlik Denetimleri ve Testleri: Yazılım geliştirme sürecinin farklı aşamalarında düzenli olarak güvenlik denetimleri ve testleri yapmak.

Penetrasyon testleri, kod incelemeleri ve güvenlik taramaları bu kapsamda kullanılabilir.

Eğitim ve Farkındalık: Geliştirici ekiplere düzenli olarak güvenlik eğitimi vermek ve güvenlik farkındalığını artırmak. Güvenlik bilincinin artırılması, ekiplerin daha güvenli tasarım ve kodlama yapmasına yardımcı olur.

Güvenlik Geri Bildirim Döngüleri: Güvenlik sorunlarını hızlı bir şekilde tespit etmek ve çözmek için etkin geri bildirim döngüleri kurmak. Geliştirme aşamalarında tespit edilen güvenlik sorunlarının hızlıca ele alınması önemlidir.

5. Security Misconfiguration

Security Misconfiguration (Güvenlik Yapılandırma Hataları), bir sistemin veya uygulamanın güvenlik ayarlarının yanlış yapılandırılması, varsayılan ayarların değiştirilmemesi veya güvenlik açısından yetersiz yapılandırılması durumunda ortaya çıkan güvenlik açıklarıdır. Bu tür hatalar, saldırganların sistemlere yetkisiz erişim sağlamasına, hassas verilere ulaşmasına veya sistemlerin bütünlüğünü bozmasına yol açabilir.

Nedenleri:

Varsayılan Ayarların Kullanılması: Yazılım ve sistemlerin varsayılan güvenlik ayarlarının değiştirilmemesi. Varsayılan kullanıcı adları, parolalar veya yapılandırma dosyaları sıkça bilinir ve saldırganlar tarafından kolayca hedef alınabilir.

Güvenlik Güncellemelerinin Yetersiz Uygulanması: Yazılım, sistem veya bileşenlerdeki güvenlik açıklarını kapatan yamaların ve güncellemelerin zamanında uygulanmaması.

Gereksiz Özelliklerin veya Hizmetlerin Etkin Olması: Kullanılmayan veya gereksiz özellikler, hizmetler veya portların açık bırakılması. Bu, saldırganlar için daha fazla saldırı yüzeyi anlamına gelir.

Yanlış Dosya ve Dizin İzinleri: Dosya ve dizinlere gereğinden fazla erişim izni verilmesi, hassas dosyaların yetkisiz kişiler tarafından okunmasına veya değiştirilmesine olanak tanır.

Detaylı Hata Mesajlarının Kullanılması: Hata mesajlarının çok detaylı olması ve sistem hakkında fazla bilgi vermesi. Bu tür bilgiler, saldırganlara sistemin yapısı hakkında bilgi sağlar.

Yanlış Yapılandırılmış Güvenlik Duvarları ve Sunucular: Güvenlik duvarlarının, web sunucularının ve diğer güvenlik cihazlarının yanlış yapılandırılması.

Nasıl önlenir:

Varsayılan Ayarların Değiştirilmesi: Varsayılan kullanıcı adları, parolalar ve yapılandırma ayarları derhal değiştirilmelidir. Varsayılan ayarların kullanılmaması, saldırganların bilinen bilgileri kullanarak sisteme erişmesini zorlaştırır.

Düzenli Güvenlik Güncellemeleri: Sistemler, yazılımlar ve tüm bileşenler düzenli olarak güncellenmeli ve güvenlik yamaları uygulanmalıdır. Otomatik güncellemeler etkinleştirilebilir ve güvenlik ihlallerine karşı hızlı yanıt verilmelidir.

Kapsamlı Yapılandırma Yönetimi: Sistemlerin ve uygulamaların güvenli yapılandırmalarının bir envanteri tutulmalı ve düzenli olarak denetlenmelidir. Gereksiz özellikler, hizmetler veya portlar devre dışı bırakılmalıdır.

Minimum Yetki İlkesinin Uygulanması: Dosya ve dizin izinleri minimum düzeyde tutulmalıdır. Sadece gerekli kullanıcılar ve süreçler, yalnızca ihtiyaç duydukları kaynaklara erişebilmelidir.

Detaylı Hata Mesajlarından Kaçınma: Hata mesajları, saldırganların sistem hakkında bilgi edinmesini zorlaştırmak için genel ve kullanıcıya yönelik bilgi vermelidir. Detaylı hata bilgileri sadece log dosyalarına kaydedilmelidir.

Güvenlik Denetimlerinin Düzenli Olarak Yapılması: Düzenli güvenlik denetimleri, yapılandırma hatalarını tespit etmek ve düzeltmek için yapılmalıdır. Bu, penetrasyon testleri, güvenlik taramaları ve konfigürasyon yönetim araçları kullanılarak gerçekleştirilebilir.

Güvenlik Politikalarının ve Standartlarının Oluşturulması: Organizasyonlar, güvenli yapılandırmalar için açık politikalar ve standartlar oluşturmalı ve bu standartlara uyumu sağlamak için düzenli eğitimler vermelidir.

Güvenlik Duvarları ve Sunucu Güvenliği: Güvenlik duvarları doğru bir şekilde yapılandırılmalı, sadece gerekli trafik izin verilmelidir. Web sunucuları ve diğer sunucular güvenli bir şekilde yapılandırılmalı ve erişim kontrolleri sıkı bir şekilde uygulanmalıdır.

6. Vulnerable and Outdated Components

Vulnerable and Outdated Components (Zayıf ve Güncel Olmayan Bileşenler), bir yazılım sisteminde kullanılan üçüncü taraf kütüphanelerin, frameworklerin, veya diğer bileşenlerin güvenlik açıklarına sahip olması veya güncellenmemiş olmasından kaynaklanan güvenlik risklerini ifade eder. Bu tür bileşenler, bilinen güvenlik açıklarına sahip olabilir ve saldırganlar tarafından hedef alınabilir, bu da sistemi çeşitli güvenlik tehditlerine açık hale getirir.

Nedenleri:

Güvenlik Güncellemelerinin Yetersiz Uygulanması: Üçüncü taraf bileşenlerdeki güvenlik açıklarını kapatan güncellemelerin veya yamaların zamanında uygulanmaması.

Kütüphanelerin veya Frameworklerin Eski Sürümlerinin Kullanılması: Yazılım geliştirme sırasında kullanılan kütüphanelerin veya frameworklerin eski sürümlerinin kullanılması ve bu bileşenlerin güvenlik açıklarına sahip olması.

Bileşenlerin Güvenlik Açıklarından Haberdar Olmama: Geliştiricilerin veya sistem yöneticilerinin kullandıkları bileşenlerdeki güvenlik açıklarından haberdar olmaması.

Güvenlik İhlallerine Yeterince Dikkat Edilmemesi: Güvenlik ihlalleri ve açıklarına dair bilgi paylaşımı ve uyarıların yeterince dikkate alınmaması.

Bağımlılık Yönetiminin Yetersiz Olması: Yazılım projelerinin bağımlılıklarının düzenli olarak gözden geçirilmemesi ve güncellenmemesi.

Nasıl önlenir:

Düzenli Güncelleme ve Yama Yönetimi: Yazılım projelerinde kullanılan tüm üçüncü taraf bileşenlerin düzenli olarak güncellenmesi. Güvenlik yamaları ve güncellemeler mümkün olduğunca hızlı bir şekilde uygulanmalıdır. Yazılım bağımlılıkları yönetim sistemleri (örneğin, npm, pip, Maven) bu süreçte yardımcı olabilir.

Bileşenlerin Güvenlik Açıklarını İzleme: Kullanılan kütüphaneler, frameworkler ve diğer bileşenler için güvenlik açıklarını izleyen araçlar kullanılmalıdır. Örneğin, OWASP Dependency-Check veya Snyk gibi araçlar, bilinen güvenlik açıklarına sahip bileşenleri tespit etmek için kullanılabilir.

Bileşen Yönetim Politikalarının Oluşturulması: Güvenli bileşen kullanımı için bir politika belirlemek ve bu politikayı uygulamak. Bu politika, hangi bileşenlerin kullanılabileceğini, güncellemelerin nasıl yönetileceğini ve güvenlik açıklarının nasıl ele alınacağını tanımlamalıdır.

Sadece Gereken Bileşenlerin Kullanımı: Gereksiz bileşenlerin kullanımı güvenlik risklerini artırabilir. Projede gerçekten ihtiyaç duyulan bileşenlerin kullanılması ve kullanılmayan bileşenlerin kaldırılması güvenlik açısından önemlidir.

Proaktif Güvenlik Testleri: Proje sırasında ve sonrasında düzenli olarak güvenlik testleri ve bağımlılık taramaları yapılmalıdır. Bu, potansiyel güvenlik açıklarını önceden tespit etmeye ve ele almaya yardımcı olur.

Geliştirici Eğitimleri: Geliştiricilere güvenli kodlama ve bileşen kullanımı konusunda düzenli eğitimler verilmeli. Bu, ekiplerin güvenlik konusunda daha bilinçli olmasını ve güvenlik risklerini minimize etmesini sağlar.

Alternatif Güvenli Bileşenlerin Seçilmesi: Zayıf veya güncel olmayan bir bileşen yerine, güvenlik açısından daha güçlü ve desteklenen bir alternatif bileşen tercih edilmelidir.

Güvenli Yapılandırma ve İzolasyon: Kullanılan bileşenlerin güvenli bir şekilde yapılandırılması ve mümkün olduğunca izole edilmesi, güvenlik açıklarının istismar edilmesini zorlaştırır.

7. Identification and Authentication Failures

Identification and Authentication Failures (Kimlik Doğrulama ve Kimlik Tespiti Hataları), bir sistemin kullanıcıların kimliklerini doğru bir şekilde tespit edememesi veya kimlik doğrulama süreçlerinin güvenli olmaması nedeniyle ortaya çıkan güvenlik açıklarını ifade eder. Bu tür hatalar, saldırganların yetkisiz erişim sağlamasına, kullanıcı hesaplarını ele geçirmesine veya kimlik doğrulama süreçlerini atlatmasına yol açabilir.

Nedenleri:

Zayıf Şifre Politikaları: Kullanıcıların zayıf veya kolay tahmin edilebilir şifreler kullanmasına izin verilmesi. Örneğin, kısa veya basit şifreler, parola tekrar kullanımı gibi durumlar.

Çok Faktörlü Kimlik Doğrulamanın (MFA) Eksikliği: Yalnızca tek faktörlü kimlik doğrulama (örneğin, yalnızca şifre) kullanılması. Bu, hesapların ele geçirilmesini kolaylaştırır.

Oturum Yönetimi Hataları: Kullanıcı oturumlarının yanlış yönetilmesi, oturum zaman aşımı ayarlarının yetersiz olması veya oturum belirteçlerinin doğru bir şekilde işlenmemesi.

Kimlik Doğrulama Bilgilerinin Yetersiz Korunması: Şifrelerin veya diğer kimlik doğrulama bilgilerinin güvenli olmayan bir şekilde saklanması (örneğin, düz metin olarak veya zayıf hash algoritmaları kullanılarak).

Açık Kimlik Doğrulama Protokollerinin Yanlış Kullanımı: Güvenli olmayan kimlik doğrulama protokollerinin kullanılması veya güvenli protokollerin yanlış yapılandırılması.

Brute Force ve Sözlük Saldırılarına Karşı Korumanın Olmaması: Kimlik doğrulama işlemleri sırasında deneme yanılma yöntemlerine (brute force) karşı önlemlerin eksik olması.

Güvenli Olmayan Kimlik Doğrulama Yöntemleri: Örneğin, sadece bir gizli soru veya kolay tahmin edilebilir biyometrik verilerle kimlik doğrulama yapılması.

Nasıl önlenir:

Güçlü Şifre Politikaları Uygulamak: Kullanıcıların güçlü ve karmaşık şifreler kullanmasını zorunlu kılmak. Şifrelerin en azından belirli bir uzunlukta olması, büyük/küçük harf, sayı ve özel karakterler içermesi sağlanmalı. Şifre tekrar kullanımını önlemek için şifre geçmişi takibi yapılmalı.

Çok Faktörlü Kimlik Doğrulama (MFA) Kullanımı: Kullanıcıların kimlik doğrulama sürecine ek bir güvenlik katmanı eklemek için MFA kullanmak. Bu, şifrelerin ele geçirilmesi durumunda bile hesap güvenliğini artırır.

Güvenli Oturum Yönetimi: Kullanıcı oturumlarını doğru bir şekilde yönetmek. Oturum belirteçleri güvenli bir şekilde saklanmalı ve işlenmeli. Oturum zaman aşımı ayarları, uzun süre kullanılmayan oturumların otomatik olarak sonlandırılmasını sağlamalı.

Kimlik Doğrulama Bilgilerinin Güvenli Saklanması: Şifrelerin hash algoritmaları (örneğin, bcrypt, Argon2) kullanılarak saklanması. Düz metin veya zayıf hash algoritmaları kullanılmamalıdır.

Kimlik Doğrulama Protokollerinin Güvenli Kullanımı: Güvenli kimlik doğrulama protokolleri (örneğin, OAuth, SAML) kullanılmalı ve doğru bir şekilde yapılandırılmalıdır. Ayrıca, HTTPS kullanılarak kimlik doğrulama işlemleri sırasında verilerin güvenli bir şekilde iletilmesi sağlanmalıdır.

Deneme Yanılma (Brute Force) Saldırılarına Karşı Koruma: Kimlik doğrulama işlemleri sırasında başarısız giriş denemelerine karşı önlemler alınmalı. Örneğin, belirli bir sayıda başarısız giriş denemesinden sonra geçici olarak hesaba erişimin engellenmesi.

Kullanıcı Eğitimleri ve Farkındalık: Kullanıcıları, güçlü şifreler oluşturma, şüpheli e-postalara dikkat etme ve kimlik avı saldırılarına karşı korunma konusunda bilinçlendirmek. Kullanıcıların kimlik doğrulama süreçlerine güvenli bir şekilde katılım göstermesi sağlanmalıdır.

Düzenli Güvenlik Denetimleri: Kimlik doğrulama ve kimlik tespiti süreçlerinin düzenli olarak denetlenmesi, güvenlik açıklarının tespit edilmesi ve hızlı bir şekilde ele alınması sağlanmalıdır.

8. Software and Data Integrity Failures

Software and Data Integrity Failures (Yazılım ve Veri Bütünlüğü Hataları), yazılım ve verilerin yetkisiz değişikliklere karşı savunmasız olması durumunu ifade eder. Bu tür hatalar, kötü amaçlı yazılımlar, zararlı kod enjeksiyonları, veri manipülasyonu veya yetkisiz erişim yoluyla sistemdeki yazılım veya verilerin bütünlüğünün bozulmasına neden olabilir. Yazılım ve veri bütünlüğü hataları, sistemin güvenilirliğini zedeler ve kullanıcılara zarar verebilir.

Nedenleri:

Yetersiz Güncelleme ve Yama Yönetimi: Yazılım ve sistem bileşenlerinin güvenlik güncellemelerinin ve yamalarının zamanında uygulanmaması. Eski yazılım sürümleri genellikle bilinen güvenlik açıklarına sahiptir.

Kötü Amaçlı Yazılım Enfeksiyonları: Sistemlerin, kötü amaçlı yazılımlara veya zararlı kodlara karşı yeterince korunmaması. Bu durum, yazılım ve verilerin yetkisiz kişiler tarafından değiştirilmesine yol açabilir.

Güvenli Olmayan Kod Depoları: Yazılım kodunun saklandığı depolarda (örneğin, Git) yeterli güvenlik önlemlerinin alınmaması. Yetkisiz kişiler bu depolara erişebilir ve kodda değişiklik yapabilir.

Güvenli Olmayan Veri Transferi: Verilerin güvenli olmayan yollarla (örneğin, şifrelenmemiş ağlar üzerinden) iletilmesi. Bu, verilerin aktarım sırasında değiştirilebileceği anlamına gelir.

Yetersiz Erişim Kontrolleri: Kullanıcıların veya sistemlerin yazılım ve verilere yetkisiz erişimini engellemek için yeterli erişim kontrollerinin olmaması. Bu, yetkisiz değişikliklere yol açabilir.

Güvenli Olmayan Üçüncü Taraf Bileşenler: Üçüncü taraf kütüphaneler ve bileşenlerin güvenli olup olmadığının kontrol edilmemesi. Zararlı kod içeren veya güvenlik açıklarına sahip bileşenler kullanılabilir.

Kod İmzasının Eksikliği: Yazılım güncellemeleri ve dosyaların orijinalliğini doğrulamak için kod imzalama mekanizmalarının kullanılmaması. Bu, kullanıcıların sahte yazılım güncellemeleri indirmesine neden olabilir.

Nasıl önlenir:

Düzenli Güncelleme ve Yama Yönetimi: Tüm yazılım ve sistem bileşenleri düzenli olarak güncellenmeli ve güvenlik yamaları uygulanmalıdır. Bu, bilinen güvenlik açıklarının kapatılmasına yardımcı olur.

Kötü Amaçlı Yazılım Koruması: Sistemlerde güncel antivirüs ve kötü amaçlı yazılım tespit araçları kullanılmalı. Ayrıca, zararlı kodların sisteme girmesini engellemek için düzenli güvenlik taramaları yapılmalıdır.

Güvenli Kod Depoları Kullanımı: Yazılım kodu depolarının güvenliğini sağlamak için erişim kontrolleri ve kimlik doğrulama mekanizmaları uygulanmalıdır. Ayrıca, kod depolarında yapılan değişiklikler denetlenmeli ve izlenmelidir.

Veri Şifreleme: Veriler, iletim sırasında ve depolama sırasında şifrelenmelidir. Güvenli iletişim protokolleri (örneğin, HTTPS, TLS) kullanılmalı ve hassas veriler güvenli bir şekilde saklanmalıdır.

Güçlü Erişim Kontrolleri: Kullanıcıların ve sistemlerin yazılım ve verilere yetkisiz erişimini engellemek için güçlü erişim kontrolleri uygulanmalıdır. Bu, rol tabanlı erişim kontrolü (RBAC) ve en az ayrıcalık ilkesi gibi yöntemlerle sağlanabilir.

Üçüncü Taraf Bileşenlerin Güvenliği: Üçüncü taraf kütüphaneler ve bileşenler kullanılırken, güvenlik kontrolleri yapılmalı ve güvenli olduklarından emin olunmalıdır. Bu bileşenlerin güvenlik güncellemeleri düzenli olarak kontrol edilmelidir.

Kod İmzası Kullanımı: Yazılım ve güncellemelerin orijinalliğini doğrulamak için kod imzalama mekanizmaları kullanılmalıdır. Bu, kullanıcıların yazılımın güvenilir bir kaynaktan geldiğini doğrulamasına yardımcı olur.

Veri Bütünlüğü Kontrolleri: Verilerin bütünlüğünü sağlamak için kriptografik hash fonksiyonları (örneğin, SHA-256) kullanılabilir. Bu, verilerin yetkisiz değişikliklere karşı korunmasına yardımcı olur.

Denetim ve İzleme: Yazılım ve veri değişikliklerinin izlenmesi ve denetlenmesi, potansiyel güvenlik ihlallerinin tespit edilmesine yardımcı olabilir. Güvenlik olayları kaydedilmeli ve düzenli olarak incelenmelidir.

Güvenlik Eğitimleri: Geliştirici ekipler ve sistem yöneticilerine güvenli yazılım geliştirme ve veri bütünlüğü konularında düzenli eğitimler verilmelidir. Bu, bilinçli kararlar alınmasına ve güvenlik uygulamalarının iyileştirilmesine katkıda bulunur.

9. Security Logging and Monitoring Failures

Security Logging and Monitoring Failures (Güvenlik Kaydı ve İzleme Hataları), bir sistemin veya uygulamanın güvenlik olaylarını yeterince kaydedememesi veya bu olayları izleyememesi durumunda ortaya çıkan güvenlik açıklarını ifade eder. Bu tür hatalar, potansiyel saldırıların ve güvenlik ihlallerinin zamanında tespit edilememesine ve uygun müdahalenin gecikmesine yol açabilir.

Nedenleri:

Yetersiz Günlük Kaydı (Logging): Sistem veya uygulamanın önemli güvenlik olaylarını yeterince kaydetmemesi. Örneğin, başarısız giriş denemeleri, yetkisiz erişim denemeleri veya sistem hatalarının kaydedilmemesi.

Yetersiz İzleme: Güvenlik olaylarının gerçek zamanlı olarak izlenmemesi. Bu, şüpheli aktivitelerin zamanında tespit edilememesine ve hızlı bir müdahale yapılamamasına yol açabilir.

Günlüklerin Yetersiz Saklanması: Güvenlik kayıtlarının yetersiz bir süre boyunca saklanması veya önemli kayıtların kaybolması. Bu, olayların geriye dönük olarak incelenmesini zorlaştırır.

Yanlış Yapılandırılmış Günlük Ayarları: Log dosyalarının yanlış yapılandırılması, önemli bilgilerin eksik kaydedilmesine neden olabilir. Örneğin, log seviyelerinin yanlış ayarlanması veya hassas bilgilerin loglara yazılması.

Yetersiz Günlük Analizi: Günlük kayıtlarının düzenli olarak analiz edilmemesi veya güvenlik olaylarının tespit edilmesi için yeterli araçların kullanılmaması.

Kritik Olayların Kaydedilmemesi: Özellikle yüksek risk taşıyan işlemler veya olaylar (örneğin, yönetici erişimi, kritik sistem değişiklikleri) için yeterli log kaydı tutulmaması.

Güvenlik Olaylarına Hızlı Müdahale Eksikliği: Güvenlik olaylarının tespit edilmesine rağmen, olaylara hızlı ve etkili bir şekilde müdahale edilmemesi.

Nasıl önlenir:

Kapsamlı Güvenlik Günlüğü Tutma: Tüm kritik güvenlik olaylarının (örneğin, başarılı/başarısız oturum açma denemeleri, veri erişim girişimleri, yapılandırma değişiklikleri) düzenli olarak kaydedilmesi sağlanmalıdır. Bu, log seviyelerinin doğru bir şekilde yapılandırılmasıyla sağlanabilir.

Gerçek Zamanlı İzleme: Güvenlik olaylarının gerçek zamanlı olarak izlenmesi için izleme araçları ve sistemleri kullanılmalıdır. SIEM (Security Information and Event Management) sistemleri bu amaçla kullanılabilir ve potansiyel tehditleri erken tespit edebilir.

Güvenlik Kayıtlarının Güvenli Saklanması: Log dosyalarının güvenli bir şekilde saklanması ve belirli bir süre boyunca (örneğin, yasal veya organizasyonel gereksinimlere göre) arşivlenmesi sağlanmalıdır. Bu, olay incelemeleri ve adli analiz için gereklidir.

Doğru Log Seviyelerinin Ayarlanması: Farklı olay türleri için uygun log seviyeleri ayarlanmalıdır. Örneğin, hata, uyarı ve bilgi düzeylerinde log kaydı yapılmalı ve kritik olaylar ayrı bir log dosyasında tutulmalıdır.

Düzenli Log Analizi: Güvenlik kayıtları düzenli olarak analiz edilmeli ve potansiyel güvenlik tehditleri için taranmalıdır. Bu analizler otomatikleştirilebilir ve anormal aktiviteleri tespit eden uyarı sistemleri kurulabilir.

Erişim Kontrolleri: Log dosyalarına erişim sıkı bir şekilde kontrol edilmelidir. Sadece yetkili kullanıcılar log dosyalarına erişim sağlayabilmeli ve log dosyaları üzerinde değişiklik yapabilmelidir.

Olay Müdahale Planları: Güvenlik olaylarına hızlı ve etkili bir şekilde müdahale etmek için bir olay müdahale planı geliştirilmelidir. Bu plan, olayların nasıl raporlanacağını, kimlerin bilgilendirileceğini ve olaylara nasıl müdahale edileceğini içermelidir.

Düzenli Güvenlik Denetimleri: Güvenlik loglarının ve izleme süreçlerinin etkinliği düzenli olarak denetlenmeli ve iyileştirme fırsatları belirlenmelidir. Denetimler, olası güvenlik açıklarını ve iyileştirilmesi gereken alanları tespit etmeye yardımcı olur.

Eğitim ve Farkındalık: Sistem yöneticileri ve güvenlik personeli, güvenlik logları ve izleme konusunda eğitilmelidir. Eğitimler, log yönetimi, izleme araçlarının kullanımı ve olay müdahale prosedürlerini kapsamalıdır.

Log İyileştirme ve Geliştirme: Sürekli olarak log yönetim ve izleme süreçlerinin iyileştirilmesi hedeflenmelidir. Teknolojik gelişmeler ve yeni tehditler göz önünde bulundurularak log yönetim sistemleri güncellenmeli ve optimize edilmelidir.

10. Server-Side Request Forgery

Server-Side Request Forgery (SSRF), bir saldırganın sunucuya, sunucunun kendisi adına başka bir kaynağa istek göndermesini sağlamasına denir. SSRF, saldırganların dahili sistemlere erişim sağlamasına, hassas bilgilere ulaşmasına veya başka güvenlik açıklarından yararlanmasına yol açabilir. Bu tür saldırılar genellikle, sunucunun bir URL veya başka bir kaynağı istemesi gereken işlevlerin istismar edilmesiyle ortaya çıkar.

Nedenleri:

Güvenilmeyen Girdilerin Yetersiz Doğrulanması: Kullanıcıların girdiği URL veya kaynak adreslerinin doğrudan sunucuya istek göndermek için kullanılması.

Saldırganlar, bu girdileri manipüle ederek sunucunun istek göndereceği hedefi kontrol edebilir.

Açık İletişim Kanalları: Sunucunun dahili ağ kaynaklarına (örneğin, localhost, özel IP aralıkları) erişimi varsa, bu tür kaynaklara istek göndermesi mümkün hale gelir. Dahili API'ler veya veritabanları gibi kaynaklara erişim sağlanabilir.

Yanlış Yapılandırılmış Güvenlik Duvarları ve Erişim Kontrolleri: Güvenlik duvarlarının ve erişim kontrollerinin yanlış yapılandırılması, sunucunun dahili ağlarda istek yapmasına izin verebilir.

Yanlış Yapılandırılmış Üçüncü Taraf Bileşenler: Üçüncü taraf kütüphanelerin veya hizmetlerin yanlış yapılandırılması sonucu, bu kütüphaneler ve hizmetler aracılığıyla isteklerin gönderilmesi. Bu durum, saldırganların bu bileşenleri kullanarak SSRF saldırıları yapmasına neden olabilir.

URL Yeniden Yönlendirmeleri ve Kısa URL'ler: URL yönlendirmelerini veya kısa URL hizmetlerini kullanarak sunucunun istemeden başka bir kaynağa istek göndermesi sağlanabilir. Bu durum, özellikle güvenilmeyen kaynaklardan gelen URL'lerin sunucu tarafından işlenmesi durumunda tehlikelidir.

Nasıl önlenir:

Girdi Doğrulama ve Filtreleme: Kullanıcı girdilerini kabul ederken sıkı doğrulama ve filtreleme yapılmalıdır. Sunucunun sadece belirli güvenilir kaynaklara istek göndermesine izin verilmelidir. URL'lerin beyaz listeye alınması ve sadece izin verilen etki alanlarına istek yapılması sağlanabilir.

Ağ Erişim Kontrolleri: Sunucunun yalnızca belirli ağlara erişebilmesi için ağ erişim kontrolleri ve güvenlik duvarları kullanılmalıdır. Özellikle, sunucunun dahili ağ kaynaklarına veya özel IP aralıklarına erişimi kısıtlanmalıdır.

İsteklerin Kontrolü ve Sınırlandırılması: Sunucu tarafından yapılan dış isteklerin sayısı ve türü sınırlandırılmalıdır. Örneğin, belirli bir süre içinde yapılan isteklerin sayısına sınır koyulabilir. Ayrıca, sunucu sadece belirli portlardan istek yapabilmelidir.

Yanıtların Doğrulanması: Sunucunun bir kaynaktan aldığı yanıtları doğrulaması ve analiz etmesi önemlidir. Yanıtların beklenen formatta ve içerikte olup olmadığı kontrol edilmelidir.

Üçüncü Taraf Bileşenlerin Güvenliği: Üçüncü taraf kütüphaneler ve hizmetlerin güvenli olduğundan emin olunmalı ve bunların yanlış yapılandırılması önlenmelidir. Bu bileşenlerin düzenli olarak güncellenmesi ve güvenlik kontrollerinin yapılması sağlanmalıdır.

Güvenlik Duvarı ve Proksi Kullanımı: Sunucu isteklerinin bir güvenlik duvarı veya proxy sunucu üzerinden geçirilmesi, isteklerin kontrol edilmesine ve potansiyel tehditlerin engellenmesine yardımcı olabilir. Bu, özellikle dahili ağlara veya hassas kaynaklara yönelik isteklerde önemlidir.

Uyarı ve İzleme Mekanizmaları: Sunucunun yaptığı istekleri izlemek ve anormal veya şüpheli aktiviteleri tespit etmek için uyarı ve izleme mekanizmaları kullanılmalıdır. Bu, potansiyel SSRF saldırılarını erken tespit etmeye ve hızlı müdahale etmeye yardımcı olabilir.

DNS Çözümleme Kısıtlamaları: Sunucunun yaptığı DNS çözümlemelerinin kontrol edilmesi ve bu çözümlemelerin yalnızca belirli güvenilir DNS sunucuları üzerinden yapılması sağlanmalıdır. Saldırganların sahte DNS sunucuları kullanarak sunucunun hedeflerini manipüle etmesi önlenebilir.