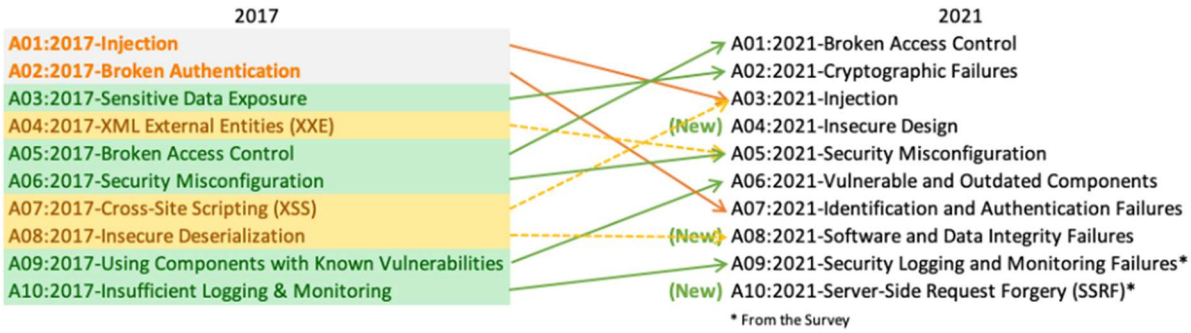
OWASP TOP 10

Owasp Top Ten

Owasp top 10, geliştiriciler ve web uygulaması güvenliği için bilgilendirici niteliğinde standart bir farkındalık belgesidir. Web uygulamalarına yönelik **en kritik güvenlik riskleri** hakkında geniş bir fikir birliğini temsil eder.

En Önemli 10 (Top 10) Web Uygulama Güvenliği Riskleri



1. Broken Access Control

Tanım ve Zaafiyet Neden Kaynaklanır

Bozuk erişim kontrolü güvenlik zaafiyeti, kullanıcıların yetkileri dışında hareket edememesi için bir politika uygular. Güvenlik zaafiyetleri genellikle yetkisiz erişim ve hassas verilerin açığa çıkması, tüm verilerin yok olmasına veya kullanıcının sınırları dışında değiştirilmesin yol açar. Yaygın 'Broken Access Control' güvenlik açıklıkları şunları içermektedir:

- En düşük ayrıcalık iklesinin ihlali, erişimin yalnızca belirli roller, kullanıcılar için verilmesi gerektiği ancak herkes tarafından kullanıma açık olduğu durumlarda.
- URL'yi (parametre kullanma veya browser'ı kurcalama), dahili uygulama durumunu veya HTML sayfasını değiştirerek, veya API isteklerini değiştiren bir saldırı aracı kullanarak erişim kontrolünü bypass etmek.
- Unique Identifier (Insecure Direct Object Reference) sağlayarak başka birinin hesabını görüntülmeye ve düzenlemeye izin vermek.
- POST, PUT, DELETE gibi erişim kontrolleri eksik olan API'lara ulaşmak ve sunucudan veri silme, yerleştirme gibi işlemlere olanak tanımak.

- Ayrıcalık yükseltme, yani oturum açmadan kullanıcı olarak hareket etme veya kullanıcı hesabıyla yönetici hesabı yetkisinde işlemler yapmak.
- Bir JSON Web Token (JWT) gibi erişim kontrol belirtecini yeniden oynatmak veyakurcalamak, veya ayrıcalık yükseltmek için JWT invalidation(JWT geçersiz kılmayı) manipüle eden bir çerez veya bir gizli dosya gibi yapılan meta data manipülasyonu.
- CORS yanlış yapılandırılması, yetkisiz/güvensiz kaynaklardan yapılan API isteklerine izin verir.

Nasıl Önlenir?

Erişim kontrolü yalnızca, saldırganın erişim kontrolünü veya meta dataları değiştiremediği güvenilir sunucu taraflı kodda ve sunucusuz API'larda etkilidir.

- Tüm kullanıcılara açık kaynaklar dışında, varsayılan olarak reddet.
- Erişim kontrol mekanizmalarını uygulayın ve Cross-Origin Resuorce Sharing (CORS) kullanımını en aza indirin.
- Model erişim kontrolleri, kullanıcının herhangi bir kaydı oluşturabileceğini, okuyabileceğini, güncelleyebileceğini veya silebileceğini kabullenmek yerine kayıt sahipliğini zorunlu kılmalı.
- Web sunucusu dizin listesini devre dışı bırakın ve dosya meta verilerinin (örn., .git, .env) ve yedekleme dosyalarının web sunucu kök dizinde bulunmadıklarından emin olun.
- Erişim kontrolü hatalarını loglayın.
- Otomatik saldırı araçlarından kaynaklanan zararı en aza indirmek için denetleyici erişimi ve API isteklerini sınırlayın.
- Oturum kapatıldıktan sonra sunucuda 'stateful' olan oturum tanımlayıcıları geçersiz kılınmalıdır. Saldırgana fırsat vermemek için JWT tokenlar kısa ömürlü olmalıdır. Daha uzun ömürlü JWT'ler için erişimi iptal etmek amacıyla kullanılan OAuth standartlarını göz önünde bulundurmaları tavsiye edilir.

2. Cryptographic Failures

Genel Bakış

Daha önceden hassas verilerin açığa çıkarılması (Sensitive data exposure) olarak bilinen, odak noktası kriptografi(veya eksikliği) ile ilgili olan hatalardır. Bunlar genellikle hassas verilerin sızdırılmasına olanak tanır. Öne çıkan CWE'ler şunlardır:

CWE-259: Use of Hard-coded Password, *CWE-327: Broken or Risky Crypto Algorithm*, and *CWE-331 Insufficient Entropy*.

Açıklama ve Daha Derin Bir Bakış

İlk olarak veriler, aktarım veya bekleme sırasındaki koruma ihtiyaçlarına göre belirlenir. Örneğin parolalar, kredi kart numaraları, sağlık kayıtları ve kişisel bilgiler ve ticari sırlar, özellikle finansal veri koruma kapsamına giriyorsa, ekstra koruma gerektirir. Bu tür tüm veriler için:

- Herhangi bir veri açık metin olarak iletiliyor mu? http,smtp, ftp gibi protokolleri ve starttls gibi tls yükseltmelerini de kapsar. Harici internet trafiği tehlikelidir. Tüm harici trafiği doğrulayın. Örneğin yük dengeleyiciler, web sunucuları veya backend sistemleri arasındaki trafiği.
- Varsayılan olarak veya eski kodlarda eski veya zayıf kriptografik algoritmalar veya protokoller uygulanıyor mu?
- Varsayılan kripto anahtarı kullanılıyor mu, zayıf kripto anahtarı üretiyor veya yeniden kullanıyor mu, yoksa uygun anahtar yönetimi veya rotasyon mu eksik? Gibi soruları sorgulamalıyız.
- Parola tabanlı anahtar türetme fonksiyonu olmadığında parolalar kriptografik anahtar olarak mu kullanılıyor?
- MD5, SHA1 gibi eski ve kullanımdan kaldırılmış hash fonksiyonları kullanılıyor mu?
    - PKCS 1 v1.5 gibi kullanımdan kaldırılmış kriptografik dolgu yöntemleri kullanılıyor mu?

Nasıl Önlenir?

- Bir uygulama tarafından iletilen, depolanan ve işlenen verileri sınıflandırın. Hangi verilerin hassas veriler olduğunu belirleyin.
- Hassas verileri saklamayın, mümkün olan en kısa sürede atın veya PCI DSS uyumlu belirteçleme veya kesme kullanın. Saklanmayan veri çalınamaz.
- Saklanan hassas verilerin şifrelendiğinden emin olun.
- Hassas verileri içeren yanıtlar için önbelleğe almayı devre dışı bırakın.
- Veri sınıflandırmasın göre gerekli güvenlik kontrollerini uygulayın.
- Argon2, scrypt, bcrypt veya PBKDF2 gibi güçlü ve salt kullanılan hash algoritmalarını kullanarak verilerinizi şifreleyin.

- Salt olarak şifreleme(decryption) yerine her zaman kimlik doğrulamalı şifrelemeyi deneyin.
- MD5, SHA1, PKCS numarası 1 v1.5 gibi kullanımdan kaldırılmış şifreleme işlevlerinden ve dolgu şemalarından kaçının.
- Yapılandırma ve ayarların etkinliğini bağımsız olarak doğrulayın.

3. Injection

Injection güvenlik açığında bir uygulama aşağıdaki güvenlik açıklıklarına karşı savunmasızdır:

- Kullanıcı tarafından sağlanan veriler, uygulama tarafında temizlenmez, filtrelenmez veya doğrulanmaz.
- Bağlama duyarlı kaçış içermeyen dinamik sorgular veya parametresiz çağrılar doğrudan yorumlayıcı(interpreter) da çalıştırılır.
- Düşmanca veriler, nesne-ilişkisel eşleme (ORM) arama parametrelerine ek, hassas kayıtları çıkarmak için kullanılır.
- Düşmanca veriler doğrudan birleştirilir veya kullanılır. SQL veya komut, dinamik sorgularda, yapılarda kötü amaçlı veriyi içerir.

Daha yaygın enjeksiyonlardan bazıları SQL, NoSQL, OS command, Object Relational Mapping(ORM), LDAP, ve Expression Langauge(EL) veya Object Graph Navigation Library(OGNL) injection. Kaynak kodlarının incelenmesi uygulamanın enjeksiyona karşı savunmasız olup olmadığını test etmenin en iyi yöntemidir. Tüm parametrelerin, header'ların, URL'lerin çerezlerin, JSON, SOAP ve XML veri girişlerinin otomatik olarak test edilmesi şiddetle tavsiye edilir.

Nasıl Önlenir?

Enjeksiyonu önlemek, verilerin, komutlardan ve sorgulardan ayrı tutulmasını gerektirir.

- Tercih edilen seçenek, tamamen yorumlayıcıyı kullanmaktan kaçınan, parametreli bir arayüz sağlayan veya Object Relational Mapping(ORM) araçlarına geçiş yapan güvenili bir API kullanmaktır.
    - Parametreli olsa bile, PL/SQL veya T-SQL sorguları veya EXECUTE IMMEDIATE, exec() gibi düşmanca verileri yürütse, yine de SQL Injection başlatabilir.
- Pozitif sunucu tarafı girdi doğrulaması kullanın. Bu tam bir savunma değildir, çünkü birçok uygulama, API'lar veya mobil uygulamalar gibi metin alanları veya özel karakterler içerir.
- Dinamik sorgular için, söz konusu yorumlayıcıya özel kaçış sözdizimini kullanarak özel karakterlerden kaçış yapın. Örneğin, tablo isimleri, sütun isimleri vb. gibi SQL yapıları kaçınılamaz ve bu nedenle kullanıcı tarafından sağlanan yapı adları tehlikelidir. Bu rapor yazma yazılımlarında yaygın bir sorundur.

- SQL enjeksiyonunda kayıların toplu olarak ifşalanması/sızdırılmasını önlemek için sorgular içerisinde LIMIT veya SQL kontrollerini kullanın.

Örnek Saldırı Senaryoları

Senaryo1#

Bir uygulama, aşağıdaki SQL çağrısının oluşturulmasında güvenilmeyen verileri kullanıyor:

String query = “SELECT * FROM accounts WHERE custID=’ ” + request.getParameter(“id”) + “ ‘ ”;

Senaryo2#

Benzer şekilde, bir uygulamanın framework’lere körü körüne güvenmesi sonucu kod tarafında böyle bir SQL güvenlik açığı ortaya çıkıyor(Örneğin Hibernate Query Language(HQL)):

Query HQLQuery = session.createQuery(“FROM accounts WHERE custID =’ ” + request.getParameter(“id”) + “ ’ ”);

Her iki durumda da saldırgan, tarayıcısındaki ‘id’ parametre değerini şu şekilde manipüle ederek kullanabilir: **‘UNION SLEEP(10);--**

http://example.com/app/accountView?id=' UNION SELECT SLEEP(10);--

Bu, her iki sorgunun anlamını, hesaplar tablosundan tüm kayıtları döndürecek şekilde değiştirir. Daha tehlikeli saldırılar verileri değiştirebilir veya silebilir veya hatta saklı prosedürleri çağırabilir.

4. Insecure Design

Tanım

Güvensiz Tasarım(Insecure Design), eksik veya etkisiz kontrol tasarımı olarak ifade edilen, farklı zayıflıkları ifade eden bir zayıflıktır. Güvensiz tasarım, ilk 10 risk kategorisinin kaynağı değildir. Güvensiz tasarım(insecure design) ile güvensiz uygulama(insecure implementation) arasında fark vardır. Güvenli bir tasarım yine de istismar edilebilecek güvenlik açıklıklarına yol açan uygulama kusurlarına sahip olabilir. Güvensiz bir tasarım, tanımı gereği, belirli saldırılara karşı savunmak için gereken güvenlik kontrollerinin asla oluşturulmaması nedeniyle mükemmel bir uygulama ile düzeltilemez. Güvensiz tasarıma katkıda bulunan faktörlerden biri, geliştirilen yazılım veya sistemde bulunan iş risk profili eksikliği ve dolayısıyla hangi düzeyde güvenlik tasarımının gerekli olduğunun belirlenememesidir.

Gereksinimler ve Kaynak Yönetimi

İşletmeyle bir uygulama için iş gereksinimlerini toplayın ve müzakere edin, buna tüm veri varlığının gizliliği, bütünlüğü ve kullanılabilirliği ve gerçekliği ile ilgili koruma gereksinimleri ve beklenen iş mantığı dahildir. Uygulamanızın ne kadar açıkta olacağını ve kiracıların ayrılmasına(erişim kontrolüne ek olarak) ihtiyacınız olup olmadığını hesaba katın. İşlevsel ve işlevsel olmayan güvenlik gereksinimleri de dahil olmak üzere teknik gereksinimleri derleyin. Güvenlik faaliyetleri dahil olmak üzere tüm tasarım, yapı, test ve işletimi kapsayan bütçeyi planlayın ve müzakere edin.

Secure Desing(Güvenli Tasarım)

Güvenli tasarım, tehditeleri sürekli olarak değerlendiren ve bilinen saldırı yöntemlerini önlemek için kodun sağlam bir şekilde tasarlanıp test edilmesini sağlayan bir kültür ve metodolojidir. Tehditin modellenmesi, iyileştirme oturumlarına (veya benzer etkinliklere) entegre edilmelidir; veri akışlarında ve erişim kontrolünde veya diğer güvenlik kontrollerinde değişiklikler arayın. Kullanıcı hikayesi geliştirmede doğru akış ve arıza durumlarını belirleyin, bunların sorumlu ve etkilenen taraflarca iyi anlaşıldığından ve üzerinde anlaşıldığından emin olun. Beklenen ve arıza akışları için varsayımları ve koşulları analiz edin, bunların hala doğru ve arzu edilir olduğundan emin olun. Varsayımların nasıl doğrulanacağını ve uygun davranışlar için gereken koşulların nasıl uygulanacağını belirleyin. Sonuçların kullanıcı hikayesinde belgelendiğinden emin olun. Hatalardan ders çıkarın ve iyileştirmeleri teşvik etmek için olumlu teşvikler sunun. Güvenli tasarım, yazılıma ekleyebileceğiniz bir eklenti veya araç değildir.

Nasıl Önlenir

- Güvenlik ve gizlilikle ilgili kontrolleri değerlendirmenize ve tasarlamanıza yardımcı olmak için AppSec profesyonelleriyle güvenli bir geliştirme yaşam döngüsü oluşturun ve kullanın
- Güvenli tasarım desenleri veya asfalt yol bileşenlerini kullanıma hazır hale getiren bir kütüphane oluşturun ve kullanın
- Kritik kimlik doğrulama, erişim denetimi, iş mantığı ve anahtar akışları için tehdit modellemesini kullanın
- Güvenlik dilini ve kontrollerini kullanıcı hikayelerine entegre edin
- Uygulamanızın her katmanına (ön uçtan arka uca) makullük kontrollerini entegre edin
- Tüm kritik akışların tehdit modeline karşı dirençli olduğunu doğrulamak için birim ve entegrasyon testleri yazın. Uygulamanızın her katmanı için kullanım durumlarını *ve yanlış kullanım durumlarını derleyin*.

- Maruziyet ve koruma ihtiyaçlarına bağlı olarak sistem ve ağ katmanlarındaki katmanları ayırın
- Kiracıları tüm katmanlarda tasarıma göre sağlam bir şekilde ayırın
- Kullanıcı veya hizmet bazında kaynak tüketimini sınırlayın

Örnek Saldırı Senaryoları

Senaryo #1: Bir kimlik bilgisi kurtarma iş akışı, NIST 800-63b, OWASP ASVS ve OWASP Top 10 tarafından yasaklanan "sorular ve cevaplar" içerebilir. Sorular ve cevaplar, birden fazla kişinin cevapları bilmesi nedeniyle kimliğin kanıtı olarak güvenilir olamaz, bu yüzden yasaktır. Bu tür kodlar kaldırılmalı ve daha güvenli bir tasarımla değiştirilmelidir.

Senaryo #2: Bir sinema zinciri grup rezervasyon indirimlerine izin verir ve depozito talep etmeden önce en fazla on beş katılımcıya sahiptir. Saldırganlar bu akışı modelleyebilir ve birkaç istekte aynı anda altı yüz koltuk ve tüm sinemaları rezerve edip edemeyeceklerini test edebilir ve bu da büyük bir gelir kaybına neden olabilir.

Senaryo #3: Bir perakende zincirinin e-ticaret web sitesi, açık artırma sitelerini yeniden satmak için üst düzey video kartları satın alan scalper'lar tarafından çalıştırılan botlara karşı korumaya sahip değildir. Bu, video kartı üreticileri ve perakende zinciri sahipleri için korkunç bir tanıtım yaratır ve bu kartları hiçbir fiyata elde edemeyen meraklılarla kötü kanlı ilişkilere neden olur. Dikkatli anti-bot tasarımı ve alan mantığı kuralları, örneğin kullanılabilir olduktan birkaç saniye sonra yapılan satın alımlar, sahte satın alımları tespit edebilir ve bu tür işlemleri reddedebilir.

5. Security Misconfiguration(Güvenlik Yanlış Yapılandırması)

Tanım

Uygulama aşağıdaki durumlarda güvenlik açığına sahip olabilir:

- Uygulama yığınının herhangi bir bölümünde uygun güvenlik güçlendirmesinin olmaması veya bulut hizmetlerinde uygunsuz şekilde yapılandırılmış izinler.
- Gereksiz özellikler etkinleştirilmiş veya yüklenmiş (örneğin, gereksiz bağlantı noktaları, hizmetler, sayfalar, hesaplar veya ayrıcalıklar).
- Varsayılan hesaplar ve şifreleri hala etkin ve değiştirilmemiş durumda.
- Hata yönetimi, yığın izlerini veya diğer aşırı bilgilendirici hata mesajlarını kullanıcılara gösterir.
- Yükseltilmiş sistemlerde en son güvenlik özellikleri devre dışıdır veya güvenli bir şekilde yapılandırılmamıştır.
- Uygulama sunucularında, uygulama çerçevelerinde (örneğin Struts, Spring, ASP.NET), kütüphanelerde, veritabanlarında vb. güvenlik ayarları güvenli değerlere ayarlanmamıştır.
- Sunucu güvenlik başlıkları veya yönergeleri göndermiyor veya bunlar güvenli değerlere ayarlanmamış.
- Yazılım güncel değil veya güvenlik açığına sahip uygulamalar.

Uyumlu ve tekrarlanabilir bir uygulama güvenliği yapılandırma süreci olmadan sistemler daha yüksek risk altındadır.

Nasıl Önlenir

Güvenli kurulum süreçleri uygulanmalıdır; bunlar şunları içerir:

- Tekrarlanabilir bir sertleştirme süreci, uygun şekilde kilitlenmiş başka bir ortamın dağıtımını hızlı ve kolay hale getirir. Geliştirme, QA ve üretim ortamlarının hepsi aynı şekilde yapılandırılmalı ve her ortamda farklı kimlik bilgileri kullanılmalıdır. Bu süreç, yeni bir güvenli ortam kurmak için gereken çabayı en aza indirmek için otomatikleştirilmelidir.
- Gereksiz özellikler, bileşenler, dokümantasyonlar ve örnekler içermeyen minimal bir platform. Kullanılmayan özellikleri ve çerçeveleri kaldırın veya yüklemeyin.
- Yama yönetimi sürecinin bir parçası olarak tüm güvenlik notlarına, güncellemelere ve yamalara uygun yapılandırmaları gözden geçirme ve güncelleme görevi (bkz. A06:2021-Güvenlik Açığı Olan ve Güncel Olmayan Bileşenler). Bulut depolama izinlerini (örneğin, S3 kova izinleri) gözden geçirin.
- Segmentlere ayrılmış bir uygulama mimarisi, segmentasyon, konteynerleştirme veya bulut güvenlik grupları (ACL'ler) ile bileşenler veya kiracılar arasında etkili ve güvenli bir ayrım sağlar.
- İstemcilere güvenlik yönergeleri gönderme, örneğin Güvenlik Başlıkları.

- Tüm ortamlarda yapılandırmaların ve ayarların etkinliğini doğrulamak için otomatik bir süreç.

Örnek Saldırı Senaryoları

Senaryo #1: Uygulama sunucusu, üretim sunucusundan kaldırılmamış örnek uygulamalarla birlikte gelir. Bu örnek uygulamalarda saldırganların sunucuyu tehlikeye atmak için kullandığı bilinen güvenlik açıkları vardır. Bu uygulamalardan birinin yönetici konsolu olduğunu ve varsayılan hesapların değiştirilmediğini varsayalım. Bu durumda saldırgan varsayılan parolalarla oturum açar ve kontrolü ele geçirir.

Senaryo #2: Dizin listeleme sunucuda devre dışı bırakılmamıştır. Bir saldırgan dizinleri listeleyebileceğini keşfeder. Saldırgan derlenmiş Java sınıflarını bulur ve indirir, bunları derler ve kodu görüntülemek için tersine mühendislik uygular. Saldırgan daha sonra uygulamada ciddi bir erişim denetimi açığı bulur.

Senaryo #3: Uygulama sunucusunun yapılandırması, ayrıntılı hata mesajlarının (örneğin, yığın izlerinin) kullanıcılara döndürülmesine izin verir. Bu, hassas bilgileri veya savunmasız olduğu bilinen bileşen sürümleri gibi temel kusurları potansiyel olarak açığa çıkarır.

Senaryo #4: Bir bulut hizmeti sağlayıcısının (CSP) varsayılan paylaşım izinleri diğer CSP kullanıcıları tarafından İnternet'e açıktır. Bu, bulut depolamada depolanan hassas verilere erişilmesine olanak tanır.

6. Vulnerable and Outdated Components

Aşağıdaki durumlarda muhtemelen savunmasızsınız:

- Kullandığınız tüm bileşenlerin sürümlerini bilmiyorsanız (hem istemci tarafı hem de sunucu tarafı). Bu, doğrudan kullandığınız bileşenlerin yanı sıra iç içe geçmiş bağımlılıkları da içerir.
- Yazılım savunmasızsa, desteklenmiyorsa veya güncel değilse. Buna işletim sistemi, web/uygulama sunucusu, veritabanı yönetim sistemi (DBMS), uygulamalar, API'ler ve tüm bileşenler, çalışma zamanı ortamları ve kitaplıklar dahildir.
- Kullandığınız bileşenlerle ilgili güvenlik bültenlerine düzenli olarak abone olmuyor ve güvenlik açıklarını düzenli olarak taramıyorsunuz.
- Altta yatan platformu, çerçeveleri ve bağımlılıkları risk tabanlı ve zamanında bir şekilde düzeltmez veya yükseltmezseniz. Bu, genellikle yama uygulamasının değişiklik kontrolü altında aylık veya üç aylık bir görev olduğu ortamlarda olur ve

kuruluşları düzeltilmiş güvenlik açıklarına günlerce veya aylarca gereksiz yere maruz kalmaya açık bırakır.

- Yazılım geliştiriciler güncellenen, yükseltilen veya yama uygulanan kütüphanelerin uyumluluğunu test etmezlerse.
- Bileşenlerin yapılandırmalarını güvence altına almazsanız (bkz. A05:2021-Güvenlik Yapılandırma Hatası).

Nasıl Önlenir

Aşağıdakileri sağlamak için bir yama yönetim süreci olmalıdır:

- Kullanılmayan bağımlılıkları, gereksiz özellikleri, bileşenleri, dosyaları ve belgeleri kaldırın.
- Sürümler, OWASP Bağımlılık Kontrolü, retire.js vb. gibi araçları kullanarak hem istemci hem de sunucu tarafı bileşenlerinin (örneğin çerçeveler, kütüphaneler) sürümlerini ve bağımlılıklarını sürekli olarak envanter edin. Bileşenlerdeki güvenlik açıkları için Ortak Güvenlik Açığı ve Maruziyetler (CVE) ve Ulusal Güvenlik Açığı Veritabanı (NVD) gibi kaynakları sürekli olarak izleyin. Süreci otomatikleştirmek için yazılım bileşim analizi araçlarını kullanın. Kullandığınız bileşenlerle ilgili güvenlik açıkları için e-posta uyarılarına abone olun.
- Bileşenleri yalnızca güvenli bağlantılar üzerinden resmi kaynaklardan edinin. Değiştirilmiş, kötü amaçlı bir bileşen içerme olasılığını azaltmak için imzalı paketleri tercih edin (Bkz. A08:2021-Yazılım ve Veri Bütünlüğü Arızaları).
- Bakımı yapılmayan veya eski sürümler için güvenlik yamaları oluşturmayan kitaplıkları ve bileşenleri izleyin. Yama yapmak mümkün değilse, keşfedilen sorunu izlemek, algılamak veya ona karşı koruma sağlamak için sanal bir yama dağıtmayı düşünün.

Her kuruluş, uygulama veya portföyün ömrü boyunca izleme, sınıflandırma ve güncelleme veya yapılandırma değişikliklerini uygulama konusunda sürekli bir plana sahip olmalıdır.

Örnek Saldırı Senaryoları

Senaryo #1: Bileşenler genellikle uygulamanın kendisiyle aynı ayrıcalıklarla çalışır, bu nedenle herhangi bir bileşendeki kusurlar ciddi etkilere yol açabilir. Bu tür kusurlar kazara (örneğin, kodlama hatası) veya kasıtlı (örneğin, bir bileşendeki arka kapı) olabilir. Keşfedilen bazı örnek istismar edilebilir bileşen güvenlik açıkları şunlardır:

- Sunucuda keyfi kod çalıştırılmasına olanak tanıyan Struts 2 uzaktan kod çalıştırma güvenlik açığı olan CVE-2017-5638, önemli ihlallerden sorumlu tutuluyor.

- Nesnelerin interneti (IoT) sıklıkla yamalanması zor veya imkansız olsa da, bunlara yama yapmanın önemi büyüktür (örneğin, biyomedikal cihazlar).

Saldırganların yamalanmamış veya yanlış yapılandırılmış sistemleri bulmasına yardımcı olan otomatik araçlar vardır. Örneğin, Shodan IoT arama motoru, Nisan 2014'te yamalanmış Heartbleed güvenlik açığından hâlâ muzdarip olan cihazları bulmanıza yardımcı olabilir.

7. Identification and Authentication Failures

Tanım

Kullanıcının kimliğinin, kimlik doğrulamasının ve oturum yönetiminin doğrulanması, kimlik doğrulamayla ilgili saldırılara karşı koruma sağlamak için kritik öneme sahiptir. Uygulama aşağıdaki durumlarda kimlik doğrulama zayıflıkları olabilir:

- Saldırganın geçerli kullanıcı adı ve parola listesine sahip olduğu kimlik bilgisi doldurma gibi otomatik saldırılara izin verir.
- Kaba kuvvet veya diğer otomatik saldırılara izin verir.
- "Password1" veya "admin/admin" gibi varsayılan, zayıf veya iyi bilinen parolalara izin verir.
- "Bilgiye dayalı yanıtlar" gibi güvenli hale getirilemeyen zayıf veya etkisiz kimlik bilgisi kurtarma ve şifre unutma süreçlerini kullanır.
- Düz metin, şifrelenmiş veya zayıf bir şekilde karıştırılmış parola veri depolarını kullanır (bkz. A02:2021-Kriptografik Arızalar).
- Çok faktörlü kimlik doğrulaması eksik veya etkisiz.
- Oturum tanımlayıcısını URL'de açığa çıkarır.
- Başarılı bir oturum açma işleminden sonra oturum tanımlayıcısını yeniden kullan.
- Oturum Kimliklerini doğru bir şekilde geçersiz kılmaz. Kullanıcı oturumları veya kimlik doğrulama belirteçleri (çoğunlukla tek oturum açma (SSO) belirteçleri) oturum kapatma veya bir süre etkin olmama sırasında doğru bir şekilde geçersiz kılınmaz.

Nasıl Önlenir

- Mümkün olduğunda, otomatik kimlik bilgisi doldurma, kaba kuvvet ve çalınan kimlik bilgisi yeniden kullanma saldırılarını önlemek için çok faktörlü kimlik doğrulamayı uygulayın.

- Özellikle yönetici kullanıcıları için varsayılan kimlik bilgileriyle gönderim veya dağıtım yapmayın.
- Yeni veya değiştirilen parolaları en kötü 10.000 parola listesine göre test etmek gibi zayıf parola kontrolleri uygulayın.
- Parola uzunluğu, karmaşıklık ve dönüşüm politikalarını Ulusal Standartlar ve Teknoloji Enstitüsü'nün (NIST) 800-63b'nin 5.1.1 bölümündeki Ezberlenmiş Sırlar yönergeleriyle veya diğer modern, kanıta dayalı parola politikalarıyla uyumlu hale getirin.
- Tüm sonuçlar için aynı mesajları kullanarak kayıt, kimlik bilgisi kurtarma ve API yollarının hesap numaralandırma saldırılarına karşı güçlendirildiğinden emin olun.
- Başarısız oturum açma girişimlerini sınırlayın veya giderek geciktirin, ancak bir hizmet reddi senaryosu oluşturmamaya dikkat edin. Tüm başarısızlıkları günlüğe kaydedin ve kimlik bilgisi doldurma, kaba kuvvet veya diğer saldırılar algılandığında yöneticileri uyarın.
- Oturum açtıktan sonra yüksek entropili yeni bir rastgele oturum kimliği üreten sunucu taraflı, güvenli, yerleşik bir oturum yöneticisi kullanın. Oturum tanımlayıcısı URL'de olmamalı, güvenli bir şekilde saklanmalı ve oturum kapatma, boşta kalma ve mutlak zaman aşımından sonra geçersiz kılınmalıdır.

Örnek Saldırı Senaryoları

Senaryo #1: Kimlik bilgisi doldurma, bilinen parolaların listelerinin kullanımı yaygın bir saldırıdır. Bir uygulamanın otomatik tehdit veya kimlik bilgisi doldurma koruması uygulamadığını varsayalım. Bu durumda, uygulama kimlik bilgilerinin geçerli olup olmadığını belirlemek için bir parola kahini olarak kullanılabilir.

Senaryo #2: Kimlik doğrulama saldırılarının çoğu, tek faktör olarak parolaların sürekli kullanımı nedeniyle gerçekleşir. Bir zamanlar en iyi uygulamalar olarak kabul edilen parola rotasyonu ve karmaşıklık gereklilikleri, kullanıcıları zayıf parolaları kullanmaya ve tekrar kullanmaya teşvik eder. Kuruluşların NIST 800-63'e göre bu uygulamaları durdurmaları ve çok faktörlü kimlik doğrulamayı kullanmaları önerilir.

Senaryo #3: Uygulama oturum zaman aşımı doğru ayarlanmamış. Bir kullanıcı bir uygulamaya erişmek için genel bir bilgisayar kullanıyor. "Çıkış" seçeneğini seçmek yerine, kullanıcı sadece tarayıcı sekmesini kapatıp uzaklaşıyor. Bir saldırgan bir saat sonra aynı tarayıcıyı kullanıyor ve kullanıcı hala doğrulanmış durumda.

8. Software and Data Integrity Failures

Tanım

Yazılım ve veri bütünlüğü hataları, bütünlük ihlallerine karşı koruma sağlamayan kod ve altyapı ile ilgilidir. Bunun bir örneği, bir uygulamanın güvenilmeyen kaynaklardan, depolarından ve içerik dağıtım ağlarından (CDN'ler) eklentilere, kitaplıklara veya modüllere güvendiği durumdur. Güvenli olmayan bir CI/CD boru hattı, yetkisiz erişim, kötü amaçlı kod veya sistem ihlali potansiyeli yaratabilir. Son olarak, birçok uygulama artık güncellemelerin yeterli bütünlük doğrulaması olmadan indirildiği ve daha önce güvenilen uygulamaya uygulandığı otomatik güncelleme işlevini içerir. Saldırganlar, dağıtılmak ve tüm kurulumlarda çalıştırılmak üzere kendi güncellemelerini yükleyebilir. Başka bir örnek ise nesnelerin veya verilerin bir saldırganın görebileceği ve değiştirebileceği bir yapıya kodlanması veya serileştirilmesidir ve güvenli olmayan serileştirmeye karşı savunmasızdır.

Nasıl Önlenir

- Yazılımın veya verilerin beklenen kaynaktan geldiğini ve değiştirilmediğini doğrulamak için dijital imzaları veya benzer mekanizmaları kullanın.
- Npm veya Maven gibi kütüphanelerin ve bağımlılıkların güvenilir depoları tükettiğinden emin olun. Daha yüksek bir risk profiliniz varsa, incelenmiş, dahili bilinen iyi bir depo barındırmayı düşünün.
- OWASP Bağımlılık Kontrolü veya OWASP CycloneDX gibi bir yazılım tedarik zinciri güvenlik aracının, bileşenlerin bilinen güvenlik açıkları içermediğini doğrulamak için kullanıldığından emin olun
- Kötü amaçlı kod veya yapılandırmanın yazılım kanalınıza sokulma olasılığını en aza indirmek için kod ve yapılandırma değişiklikleri için bir inceleme süreci olduğundan emin olun.
- Derleme ve dağıtım süreçlerinde akan kodun bütünlüğünü garanti altına almak için CI/CD işlem hattınızın uygun ayırma, yapılandırma ve erişim denetimine sahip olduğundan emin olun.
- İmzalanmamış veya şifrelenmemiş serileştirilmiş verilerin, kurcalanmayı veya serileştirilmiş verilerin tekrar oynatılmasını tespit etmek için bir tür bütünlük kontrolü veya dijital imza olmadan güvenilmeyen istemcilere gönderilmediğinden emin olun.

Örnek Saldırı Senaryoları

Senaryo #1 İmzalamadan güncelleme: Birçok ev yönlendiricisi, set üstü kutu, cihaz yazılımı ve diğerleri imzalanmış yazılım aracılığıyla güncellemeleri doğrulamaz. İmzalanmamış yazılım saldırganlar için büyüyen bir hedeftir ve daha da kötüleşmesi

beklenmektedir. Bu büyük bir endişe kaynağıdır çünkü çoğu zaman gelecekteki bir sürümde düzeltmek ve önceki sürümlerin eskimesini beklemek dışında bir düzeltme mekanizması yoktur.

Senaryo #2 SolarWinds kötü amaçlı güncelleme : Ulus devletlerin güncelleme mekanizmalarına saldırdığı biliniyor, son zamanlardaki dikkat çekici saldırılardan biri SolarWinds Orion saldırısıydı. Yazılımı geliştiren şirketin güvenli derleme ve güncelleme bütünlüğü süreçleri vardı. Yine de bunlar altüst edilebildi ve şirket birkaç ay boyunca 18.000'den fazla kuruluşa oldukça hedefli kötü amaçlı bir güncelleme dağıttı, bunların yaklaşık 100'ü etkilendi. Bu, tarihin bu doğadaki en kapsamlı ve en önemli ihlallerinden biridir.

Senaryo #3 Güvensiz Deserializasyon: Bir React uygulaması bir dizi Spring Boot mikro hizmeti çağırır. İşlevsel programcılar oldukları için kodlarının değişmez olduğundan emin olmaya çalıştılar. Buldukları çözüm, kullanıcı durumunu serileştirmek ve her istekte ileri geri iletmektir. Bir saldırgan "rO0" Java nesne imzasını (base64'te) fark eder ve uygulama sunucusunda uzaktan kod yürütme elde etmek için Java Serial Killer aracını kullanır.

9. Security Logging and Monitoring Failures

Tanım

OWASP Top 10 2021'e geri dönersek, bu kategori aktif ihlalleri tespit etmeye, yükseltmeye ve bunlara yanıt vermeye yardımcı olmak içindir. Kayıt tutma ve izleme olmadan ihlaller tespit edilemez. Yetersiz kayıt tutma, tespit etme, izleme ve aktif yanıt her zaman meydana gelir:

- Oturum açma, başarısız oturum açma ve yüksek değerli işlemler gibi denetlenebilir olaylar günlüğe kaydedilmez.
- Uyarılar ve hatalar, hayır, yetersiz veya belirsiz günlük mesajları üretir.
- Uygulama ve API'lerin günlükleri şüpheli etkinliklere karşı izlenmez.
- Günlükler yalnızca yerel olarak saklanır.
- Uygun uyarı eşikleri ve yanıt yükseltme süreçleri mevcut değil veya etkili değil.
- Dinamik uygulama güvenliği testi (DAST) araçları (OWASP ZAP gibi) tarafından yapılan penetrasyon testleri ve taramalar uyarıları tetiklemez.
- Uygulama, gerçek zamanlı veya gerçek zamana yakın bir zamanda etkin saldırıları algılayamaz, iletemez veya uyarı veremez.

Günlük kaydı ve uyarı olaylarını bir kullanıcıya veya saldırgana görünür hale getirerek bilgi sızdırılmasına karşı savunmasız hale getirirsiniz.

Nasıl Önlenir

Geliştiriciler, uygulamanın riskine bağlı olarak aşağıdaki denetimlerin bir kısmını veya tamamını uygulamalıdır:

- Tüm oturum açma, erişim kontrolü ve sunucu tarafı giriş doğrulama hatalarının, şüpheli veya kötü niyetli hesapları belirlemek için yeterli kullanıcı bağlamıyla birlikte kaydedilebildiğinden ve gecikmeli adli analiz yapılmasına olanak sağlamak için yeterli süre boyunca saklanabildiğinden emin olun.
- Günlüklerin, günlük yönetimi çözümlerinin kolayca kullanabileceği bir biçimde oluşturulduğundan emin olun.
- Kayıt veya izleme sistemlerine yönelik enjeksiyonları veya saldırıları önlemek için kayıt verilerinin doğru şekilde kodlandığından emin olun.
- Yüksek değerli işlemlerin, yalnızca ekleme yapılabilen veritabanı tabloları veya benzeri gibi, kurcalanmayı veya silinmeyi önlemek için bütünlük kontrollerine sahip bir denetim izine sahip olduğundan emin olun.
- DevSecOps ekipleri, şüpheli faaliyetlerin tespit edilip hızla yanıtlanabilmesi için etkili izleme ve uyarı sistemleri kurmalıdır.
- Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) 800-61r2 veya sonraki bir sürüm gibi bir olay yanıtlama ve kurtarma planı oluşturun veya benimseyin.

OWASP ModSecurity Core Rule Set gibi ticari ve açık kaynaklı uygulama koruma çerçeveleri ve Elasticsearch, Logstash, Kibana (ELK) yığını gibi özel panolar ve uyarılar içeren açık kaynaklı günlük ilişkilendirme yazılımları mevcuttur.

Örnek Saldırı Senaryoları

Senaryo #1: Bir çocuk sağlık planı sağlayıcısının web sitesi operatörü, izleme ve kayıt eksikliği nedeniyle bir ihlali tespit edemedi. Harici bir taraf, sağlık planı sağlayıcısına bir saldırganın 3,5 milyondan fazla çocuğun binlerce hassas sağlık kaydına eriştiğini ve bunları değiştirdiğini bildirdi. Olay sonrası yapılan bir inceleme, web sitesi geliştiricilerinin önemli güvenlik açıklarını gidermediğini buldu. Sistemde kayıt veya izleme olmadığından, veri ihlali 2013'ten beri, yani yedi yıldan uzun bir süreden beri devam ediyor olabilir.

Senaryo #2: Büyük bir Hint havayolunda, milyonlarca yolcunun pasaport ve kredi kartı verileri de dahil olmak üzere on yıldan fazla kişisel verisini içeren bir veri ihlali yaşandı. Veri ihlali, bir süre sonra havayolunu ihlalden haberdar eden üçüncü taraf bir bulut barındırma sağlayıcısında gerçekleşti.

Senaryo #3: Büyük bir Avrupa havayolu şirketi GDPR'de bildirilmesi gereken bir ihlal yaşadı. İhlalin, saldırganların 400.000'den fazla müşteri ödeme kaydını toplayarak istismar ettiği ödeme uygulaması güvenlik açıklarından kaynaklandığı bildirildi. Bunun sonucunda havayoluna gizlilik düzenleyicisi tarafından 20 milyon pound para cezası kesildi.

10.Server Side Request Forgery(SSRF)

Tanım

SSRF kusurları, bir web uygulaması kullanıcı tarafından sağlanan URL'yi doğrulamadan uzak bir kaynağı aldığında ortaya çıkar. Bir saldırganın, bir güvenlik duvarı, VPN veya başka bir tür ağ erişim kontrol listesi (ACL) tarafından korunsa bile, uygulamayı beklenmedik bir hedefe hazırlanmış bir istek göndermeye zorlamasını sağlar.

Modern web uygulamaları son kullanıcılara kullanışlı özellikler sağladıkça, bir URL almak yaygın bir senaryo haline geliyor. Sonuç olarak, SSRF'nin görülme sıklığı artıyor. Ayrıca, bulut hizmetleri ve mimarilerin karmaşıklığı nedeniyle SSRF'nin ciddiyeti daha da artıyor.

Nasıl Önlenir

Geliştiriciler, aşağıdaki derinlemesine savunma kontrollerinin bir kısmını veya tamamını uygulayarak SSRF'yi önleyebilirler:

Ağ katmanından (OSI Network Layer)

- SSRF'nin etkisini azaltmak için uzak kaynak erişim işlevselliğini ayrı ağlara ayırın
- Tüm temel intranet trafiğini engellemek için "varsayılan olarak reddet" güvenlik duvarı politikalarını veya ağ erişim kontrol kurallarını uygulayın.
  İpuçları:
  ~ Uygulamalara dayalı güvenlik duvarı kuralları için bir sahiplik ve yaşam döngüsü oluşturun.

  ~ Güvenlik duvarlarında kabul edilen *ve*
  engellenen tüm ağ akışlarını günlüğe kaydedin.

Uygulama katmanından(OSI Application Layer):

- Müşteri tarafından sağlanan tüm giriş verilerini temizleyin ve doğrulayın
- URL şemasını, bağlantı noktasını ve hedefi olumlu bir izin listesiyle zorunlu kılın
- Müşterilere ham yanıtlar göndermeyin
- HTTP yönlendirmelerini devre dışı bırak
- DNS yeniden bağlama ve "kontrol zamanı, kullanım zamanı" (TOCTOU) yarış koşulları gibi saldırılardan kaçınmak için URL tutarlılığına dikkat edin

SSRF'yi reddetme listesi veya düzenli ifade kullanarak hafifletmeyin. Saldırganlar reddetme listelerini atlatmak için yük listelerine, araçlara ve becerilere sahiptir.

Dikkate alınması gereken ek önlemler:

- Ön sistemlerde (örneğin OpenID) diğer güvenlikle ilgili servisleri dağıtmayın. Bu sistemlerdeki yerel trafiği kontrol edin (örneğin localhost)
- Adanmış ve yönetilebilir kullanıcı gruplarına sahip ön uçlar için, çok yüksek koruma gereksinimlerini göz önünde bulundurmak amacıyla bağımsız sistemlerde ağ şifrelemesi (örneğin VPN'ler) kullanın

Örnek Saldırı Senaryoları

Saldırganlar, web uygulama güvenlik duvarları, güvenlik duvarları veya ağ ACL'leri arkasında korunan sistemlere saldırmak için SSRF'yi şu gibi senaryoları kullanarak kullanabilirler:

Senaryo #1: Dahili sunucularda port taraması – Ağ mimarisi bölümlendirilmemişse, saldırganlar dahili ağları haritalayabilir ve bağlantı sonuçlarından veya bağlanmak için geçen süreden dahili sunuculardaki portların açık mı kapalı mı olduğunu belirleyebilir veya SSRF yük bağlantılarını reddedebilir.

Senaryo #2:file:///etc/passwd Hassas verilerin ifşa edilmesi – Saldırganlar, ve gibi hassas bilgileri elde etmek için yerel dosyalara veya dahili hizmetlere erişebilir http://localhost:28017/.

Senaryo #3: Bulut hizmetlerinin meta veri depolamasına erişim – Çoğu bulut sağlayıcısının . gibi meta veri depolaması vardır http://169.254.169.254/. Bir saldırgan hassas bilgileri elde etmek için meta verileri okuyabilir.

Senaryo #4: Dahili hizmetlerin tehlikeye atılması – Saldırgan, Uzaktan Kod Yürütme (RCE) veya Hizmet Reddi (DoS) gibi daha fazla saldırı gerçekleştirmek için dahili hizmetleri kötüye kullanabilir.