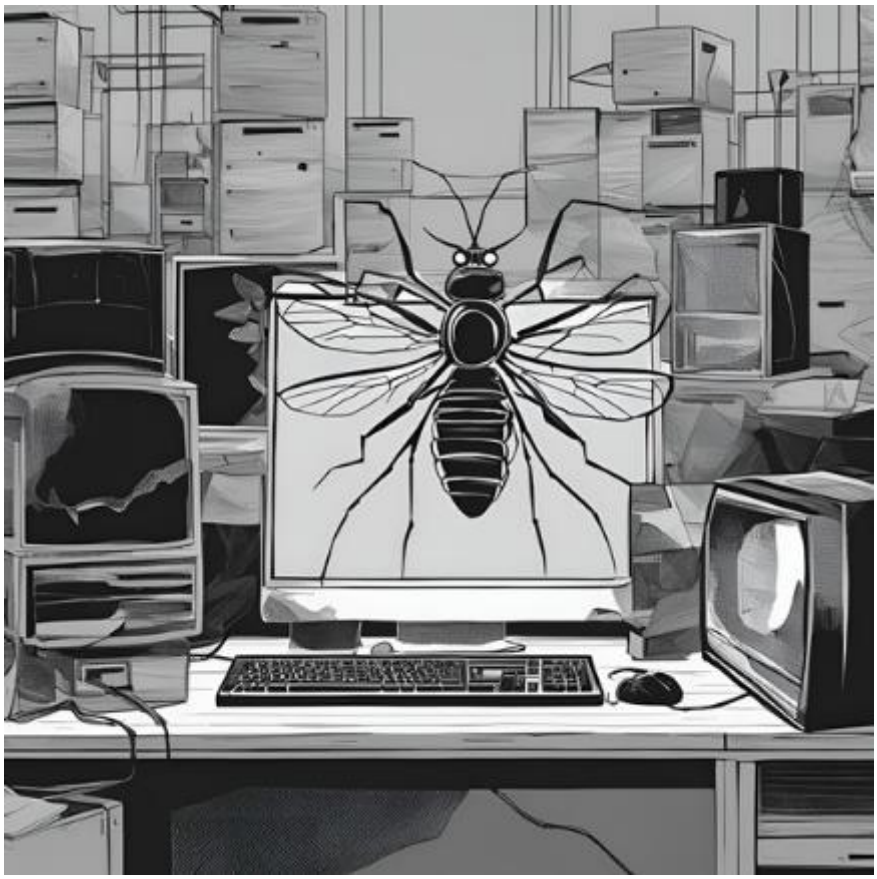

OWASP Top 10 API Security Risks – 2023

DUYGU KAÇAR

12.07.2024

İçindekiler

API1:2023- Broken Object Level Authorization

API'lerde sıkça görülen bir güvenlik zafiyetidir. Bu zafiyet, bir kullanıcının veya istemcinin, uygun yetkilendirme kontrolleri olmadan başka bir kullanıcının kaynaklarına erişmesine olanak tanır. BOLA, özellikle RESTful API'lerde yaygındır ve genellikle yetersiz veya yanlış yapılandırılmış erişim kontrolleri nedeniyle ortaya çıkar.

KORUNMA YÖNTEMLERİ:

1. **Erişim Kontrollerini Uygulamak:**
    - Her API uç noktası için sıkı erişim kontrolleri uygulayın.
    - Kullanıcıların yalnızca kendi verilerine erişimini sağlamak için yetkilendirme kontrolleri ekleyin.
2. **ID'leri Gizlemek:**
    - Hassas nesne tanımlayıcılarını (ID) doğrudan kullanmak yerine, bunları gizlemek veya maskelenmiş formlarını kullanmak.
3. **Doğrulama ve Yetkilendirme:**
    - API'ye yapılan her istekte kimlik doğrulama ve yetkilendirme mekanizmalarını zorunlu kılın.
    - Role-based access control (RBAC) veya attribute-based access control (ABAC) kullanarak erişim politikalarını tanımlayın.
4. **Güvenlik Testleri:**
    - Düzenli olarak güvenlik testleri yapın, güvenlik açıklarını tespit etmek için penetration testleri ve otomatik araçlar kullanın.
5. **Günlükleme ve İzleme:**
    - API erişimlerini ve yetkilendirme kontrollerini izlemek için kapsamlı günlükleme ve izleme mekanizmaları kullanın.
    - Anormal erişim veya yetkilendirme hatalarını tespit etmek için bu günlükleri düzenli olarak analiz edin.
6. **Güvenlik Duvarları ve Gatewayler:**
    - Web Application Firewall (WAF) ve API Gateway'leri kullanarak API'lere yapılan istekleri filtreleyin ve güvenlik politikalarını zorunlu kılın.

Bu yöntemler, Broken Object Level Authorization zafiyetini azaltmaya ve API güvenliğini artırmaya yardımcı olacaktır.

API2:2023- Broken Authentication

API'lerde kimlik doğrulama mekanizmalarının yanlış veya yetersiz şekilde uygulanmasından kaynaklanan bir güvenlik zafiyetidir. Bu zafiyet, saldırganların kullanıcı hesaplarına yetkisiz erişim elde etmesine neden olabilir. Broken Authentication zafiyetleri, genellikle zayıf parola politikaları, oturum yönetimi hataları ve güvenli olmayan kimlik doğrulama süreçlerinden kaynaklanır.

KORUNMA YÖNTEMLERİ:

1. **Güçlü Parola Politikaları:**
    - Kullanıcıların güçlü ve benzersiz parolalar kullanmasını zorunlu kılın.
    - Parola karmaşıklaştırma kuralları belirleyin (en az 8 karakter, büyük/küçük harf, rakam ve özel karakter içermesi gibi).
    - Parola tekrar kullanımını önleyin ve düzenli olarak parola değiştirmeyi teşvik edin.
2. **Çok Faktörlü Kimlik Doğrulama (MFA):**
    - Kullanıcı oturum açma işlemlerinde çok faktörlü kimlik doğrulama (MFA) kullanın.
    - SMS, e-posta veya özel kimlik doğrulama uygulamaları gibi ek doğrulama yöntemleri uygulayın.
3. **Güvenli Oturum Yönetimi:**
    - Oturum kimliklerini (session IDs) tahmin edilemez ve güvenli yapın.
    - Oturum kimliklerinin çalınmasını önlemek için HTTPS kullanın ve oturum kimliklerinin URL'lerde taşınmasını engelleyin.
    - Kullanıcı oturumlarını zaman aşımına uğratın ve etkin olmayan oturumları otomatik olarak sonlandırın.
4. **Parola Saklama ve Şifreleme:**
    - Kullanıcı parolalarını saklarken güçlü şifreleme algoritmaları (örn. bcrypt, scrypt) kullanın.
    - Parola doğrulama işlemleri sırasında salt (random salt) ekleyerek hash fonksiyonları uygulayın.
5. **Brute Force ve Rate Limiting Önlemleri:**
    - Brute force saldırılarına karşı koruma sağlamak için giriş denemelerini sınırlayın (rate limiting).

- Belirli sayıda başarısız giriş denemesinden sonra kullanıcı hesaplarını geçici olarak kilitleyin.

6. **Düzenli Güvenlik Denetimleri:**
   - Kimlik doğrulama mekanizmalarını düzenli olarak test edin ve güvenlik açıklarını belirlemek için penetrasyon testleri yapın.
   - Zayıf noktaları ve güvenlik açıklarını belirlemek için otomatik güvenlik araçları kullanın.
7. **Güvenlik Güncellemeleri ve Yamalar:**
   - API kimlik doğrulama bileşenleri ve kütüphaneleri için güncellemeleri ve yamaları düzenli olarak uygulayın.
   - Bilinen güvenlik açıklarına karşı sürekli olarak güncel kalın.

Bu yöntemler, Broken Authentication zafiyetlerini önlemeye ve API kimlik doğrulama süreçlerini daha güvenli hale getirmeye yardımcı olacaktır.

API3:2023- Broken Object Property Level Authorization

OWASP API Security Top 10 listesinde yer alan bir güvenlik açığı türüdür. Bu açık, API'nin bireysel objelerin veya özelliklerin yetkilendirilmesinde yetersiz kalması durumunda ortaya çıkar. Bu da kötü niyetli kullanıcıların, sahip olmamaları gereken verilere erişebilmesine veya bunları değiştirebilmesine olanak tanır.

BROKEN OBJECT PROPERTY LEVEL AUTHORİZATİON NEDİR?

- **Tanım:** API, belirli bir objeye veya objenin belirli bir özelliğine erişimi veya bu özelliği değiştirmeyi doğru şekilde yetkilendirmediğinde oluşur. Örneğin, bir kullanıcı yalnızca kendi hesap bilgilerini görüntüleyebilmeliyken, başka bir kullanıcının hesap bilgilerine erişebilmesi bu tür bir güvenlik açığına örnektir.

- **Riskler**: Kullanıcılar, sahip olmamaları gereken verilere erişebilir, hassas bilgileri ifşa edebilir veya kritik sistem değişiklikleri yapabilirler.

KORUNMA YÖNTEMLERİ

1. Yetkilendirme Kontrolleri Uygulama:

- Her bir API isteğinde yetkilendirme kontrollerinin kapsamlı ve tutarlı bir şekilde uygulanmasını sağlamak.

- Kullanıcıların yalnızca kendi verilerine veya yetkili oldukları verilere erişmesini sağlayacak rol tabanlı erişim kontrolleri (RBAC) ve detaylı yetkilendirme mekanizmaları uygulamak.

2. Veri Doğrulama ve Filtreleme:

- API'ye gelen tüm verileri doğrulamak ve gerektiğinde filtrelemek.

- Hassas bilgileri gizlemek veya maskelemek.

3. Detaylı Denetim ve Günlükleme:

- API erişimlerini ve yapılan işlemleri detaylı bir şekilde günlüklemek.

- Anormal erişim ve kullanım modellerini tespit etmek için log analiz araçlarını kullanmak.

4. En Az Yetki İlkesi (Principle of Least Privilege):

- Kullanıcı ve servis hesaplarına sadece gerekli olan en düşük yetkilerin verilmesini sağlamak.

- Gereksiz veya aşırı yetkilerin verilmesini önlemek.

5. API Güvenlik Testleri ve Denetimler:

- Düzenli olarak API güvenlik testleri yapmak ve yetkilendirme kontrollerinin sağlamlığını test etmek.

- Güvenlik açıklarını tespit etmek için otomatik araçlar ve manuel penetrasyon testleri kullanmak.

6. Güvenli Yazılım Geliştirme Pratikleri:

- Geliştirme sürecinde güvenlik en iyi uygulamalarını takip etmek.

- OWASP ve benzeri güvenlik topluluklarının tavsiyelerini dikkate almak.

Bu yöntemler, API'lerdeki yetkilendirme açıklarını önlemek ve güvenli bir API ortamı oluşturmak için önemli adımlardır.

API4:2023- Unrestricted Resource Consumption

OWASP API Security Top 10 listesinde yer alan bir diğer güvenlik açığı türüdür. Bu açık, bir API'nin kaynak tüketimini kısıtlayamaması durumunda ortaya çıkar. Bu da API'nin hizmet dışı kalmasına veya kaynaklarının kötüye kullanılmasına yol açabilir.

UNRESTRİCTED RESOURCE CONSUMPTİON NEDİR?

- Tanım: API'lerin kaynak tüketimini (CPU, bellek, bant genişliği, disk alanı gibi) kısıtlamaması veya yönetememesi durumunda oluşur. Kötü niyetli kullanıcılar veya

hatalı uygulamalar bu açıkları kullanarak API'yi aşırı yükleyebilir ve bu da hizmet kesintilerine veya performans düşüşlerine yol açabilir.

- Riskler: Denial of Service (DoS) saldırılarına maruz kalma, aşırı kaynak tüketimi nedeniyle yüksek maliyetler, performans sorunları ve kullanıcı memnuniyetsizliği.

KORUNMA YÖNTEMLERİ

1. Rate Limiting ve Throttling Uygulama:

- API isteklerinin sayısını belirli bir süre zarfında sınırlandırmak (rate limiting).

- Bir kullanıcının belirli bir zaman diliminde yapabileceği istek sayısını kısıtlamak (throttling).

- Örneğin, bir IP adresi veya API anahtarı başına saniyede belirli bir sayıda istek izin vermek.

2. Kaynak Tüketimi İzleme ve Yönetme:

- API'nin kaynak tüketimini sürekli izlemek ve aşırı tüketimi tespit etmek için izleme araçları kullanmak.

- Aşırı kaynak tüketimi durumunda uyarılar oluşturmak ve otomatik aksiyonlar almak.

3. Kota ve Bütçe Tanımlama:

- Kullanıcılar veya uygulamalar için kaynak kullanım kotaları tanımlamak.

- Belirli bir süre içinde kullanılabilecek maksimum kaynak miktarını belirlemek.

- Kullanım kotalarını aşan istekleri reddetmek veya sınırlamak.

- API altyapısını, trafik artışlarına otomatik olarak ölçeklendirecek şekilde yapılandırmak.

- Yük dengeleme ve yedekleme mekanizmaları ile kaynak kullanımını dağıtmak.

5. Güvenli Yazılım Geliştirme Pratikleri:

- API geliştirirken, her isteğin ve yanıtın kaynak tüketimini dikkate almak.

- Büyük veri setlerinin işlenmesi veya döndürülmesi gereken durumlarda, sonuçları parçalar halinde döndürmek veya sayfalama kullanmak.

6. Anomali Tespiti ve Yanıt Verme:

- Normal kullanım dışı kaynak tüketim paternlerini tespit etmek için anomali tespit sistemleri kullanmak.

- Olağandışı kaynak tüketimi tespit edildiğinde otomatik olarak yanıt

API5:2023 - Broken Function Level Authorization

OWASP API Security Top 10 listesinde yer alan bir güvenlik açığı türüdür. Bu açık, API'nin işlev seviyesinde yetkilendirme kontrollerini doğru bir şekilde uygulamaması durumunda ortaya çıkar. Bu, kötü niyetli kullanıcıların, sahip olmamaları gereken işlevlere erişmesine veya bu işlevleri kullanmasına olanak tanır.

BROKEN FUNCTİON LEVEL AUTHORİZATİON NEDİR?

- Tanım: API, belirli bir işlevin veya metodun yetkilendirme kontrollerini doğru şekilde uygulamadığında meydana gelir. Örneğin, bir kullanıcı admin yetkisine sahip olmasa da, admin işlevlerini çağırarak yönetici işlemleri gerçekleştirebilir.

- Riskler: Kullanıcılar, yetkileri olmadan kritik işlemler yapabilir, hassas bilgileri değiştirebilir veya sistemin işleyişini bozabilir.

KORUNMA YÖNTEMLERİ

1. Rol Tabanlı Erişim Kontrolü (RBAC) Uygulama:

- Kullanıcı rollerine dayalı olarak işlev erişimini sınırlamak.

- Her kullanıcıya yalnızca gerektiği kadar yetki vermek ve her işlev için hangi rollerin yetkili olduğunu açıkça belirlemek.

2. Yetkilendirme Kontrollerini Merkezi Olarak Yönetme:

- Yetkilendirme kontrollerini merkezi bir şekilde yönetmek ve tutarlı bir şekilde uygulamak.

- Tüm API uç noktalarında aynı yetkilendirme politikalarının uygulanmasını sağlamak.

3. Detaylı Günlükleme ve İzleme:

- Kullanıcı aktivitelerini detaylı bir şekilde günlüklemek ve izlemek.

- Yetkisiz erişim veya yetkisiz işlev çağrılarını tespit etmek ve bu durumlara hızlıca yanıt vermek.

4. Güvenlik Testleri ve Denetimler:

- Düzenli olarak güvenlik testleri ve denetimler yapmak.

- Yetkilendirme kontrollerinin sağlamlığını test etmek ve olası açıkları belirlemek için penetrasyon testleri ve otomatik güvenlik tarama araçları kullanmak.

5. En Az Yetki İlkesi (Principle of Least Privilege):

- Kullanıcı ve servis hesaplarına sadece gerekli olan en düşük yetkilerin verilmesini sağlamak.

- Gereksiz veya aşırı yetkilerin verilmesini önlemek.

6. API Gateway ve WAF Kullanma:

- API gateway'ler ve Web Application Firewall (WAF) kullanarak işlev seviyesinde yetkilendirme kontrollerini güçlendirmek.

- API çağrılarının yetkilendirme politikalarına uygun olup olmadığını doğrulamak.

7. Güvenli Yazılım Geliştirme Pratikleri:

- Geliştirme sürecinde güvenlik en iyi uygulamalarını takip etmek.

- OWASP ve benzeri güvenlik topluluklarının tavsiyelerini dikkate almak.

Bu yöntemler, API'lerdeki işlev seviyesindeki yetkilendirme açıklarını önlemek ve güvenli bir API ortamı oluşturmak için önemlidir.

API6:2023 - Unrestricted Access to Sensitive Business Flows

OWASP API Security Top 10 listesinde yer alan bir güvenlik açığı türüdür. Bu açık, API'lerin kritik iş süreçlerine sınırsız erişim sağlaması ve bu süreçlerin yeterince korunmaması durumunda ortaya çıkar. Bu da kötü niyetli kullanıcıların, önemli iş süreçlerini kötüye kullanmasına veya hassas işlemleri gerçekleştirmesine olanak tanır.

UNRESTRİCTED ACCESS TO SENSİTİVE BUSİNESS FLOWS NEDİR?

- TanımAPI, kritik iş süreçlerine veya hassas iş akışlarına yönelik yetkilendirme kontrollerini yeterince sağlamadığında meydana gelir. Örneğin, bir kullanıcı, doğrulama veya yetkilendirme olmadan ödeme işlemlerini başlatabilir veya değiştirebilir.

- Riskler:Finansal kayıplar, iş süreçlerinin bozulması, müşteri güveninin sarsılması ve yasal yükümlülükler.

KORUNMA YÖNTEMLERİ

1. İş Süreçleri İçin Güçlü Yetkilendirme Kontrolleri Uygulama:

- Hassas iş süreçlerine erişim sağlanmadan önce kullanıcıların yetkilerini doğrulamak.

- Özellikle kritik iş akışları için çok faktörlü kimlik doğrulama (MFA) ve detaylı yetkilendirme politikaları uygulamak.

2. İş Süreçlerini Kategorize Etme ve Koruma:

- Tüm iş süreçlerini önem derecesine göre kategorize etmek ve hassas iş süreçlerini belirlemek.

- Bu hassas iş süreçleri için ek güvenlik kontrolleri uygulamak.

3. Kapsamlı Günlükleme ve İzleme:

- Tüm hassas iş akışlarına yönelik işlemleri detaylı bir şekilde günlüklemek ve izlemek.

- Anormal veya yetkisiz erişim denemelerini tespit etmek ve bu durumlara hızlıca yanıt vermek.

4. Güvenlik Testleri ve Denetimler:

- Düzenli olarak güvenlik testleri ve denetimler yapmak.

- Hassas iş süreçlerine yönelik yetkilendirme kontrollerinin sağlamlığını test etmek ve olası açıkları belirlemek için penetrasyon testleri ve otomatik güvenlik tarama araçları kullanmak.

5. Veri Maskeleme ve Şifreleme:

- Hassas iş süreçleri sırasında işlenen verileri maskelemek veya şifrelemek.

- Hassas verilerin yetkisiz kullanıcılar tarafından görülmesini veya değiştirilmesini önlemek.

6. API Gateway ve WAF Kullanma:

- API gateway'ler ve Web Application Firewall (WAF) kullanarak hassas iş süreçlerine erişimi korumak.

- API çağrılarının yetkilendirme politikalarına uygun olup olmadığını doğrulamak ve anormal aktiviteleri engellemek.

7. Güvenli Yazılım Geliştirme Pratikleri:

- Geliştirme sürecinde güvenlik en iyi uygulamalarını takip etmek.

- OWASP ve benzeri güvenlik topluluklarının tavsiyelerini dikkate almak.

Bu yöntemler, API'lerdeki hassas iş süreçlerine sınırsız erişim açıklarını önlemek ve güvenli bir API ortamı oluşturmak için önemlidir.

API7:2023 - Server Side Request Forgery (SSRF)

OWASP API Security Top 10 listesinde yer alan ve API'lerin saldırıya maruz kalabileceği önemli bir güvenlik açığıdır. SSRF açıkları, saldırganların API sunucusunu, saldırganın kontrol ettiği veya seçtiği bir kaynağa istek göndermesi için kandırmasına olanak tanır. Bu tür saldırılar, saldırganların iç ağlara erişimini, hassas bilgilere ulaşmasını veya diğer sistemlere saldırı başlatmasını sağlar.

SERVER SİDE REQUEST FORGERY (SSRF) NEDİR?

- Tanım:SSRF, bir saldırganın, hedef sunucunun HTTP isteklerini kötüye kullanarak, sunucunun iç ağdaki veya internetteki diğer kaynaklara istek göndermesini sağlamasıdır. Bu saldırılar, güvenlik duvarlarının ve diğer güvenlik kontrollerinin arkasındaki sistemlere erişim sağlamaya çalışır.

- Riskler:İç ağ hizmetlerine erişim, hassas bilgilerin ifşası, veritabanı sunucularına saldırılar, diğer sistemlere yönelik ek saldırılar ve sunucuların kaynaklarının kötüye kullanılması.

KORUNMA YÖNTEMLERİ

1. Dışa Dönük İstekleri Sınırlama:

- API sunucusunun dış kaynaklara istek göndermesini gerektirmeyen durumlarda bu yeteneği tamamen devre dışı bırakmak.

- Sunucunun yalnızca güvenilir ve beklenen dış kaynaklara istek göndermesine izin vermek.

2. Güvenlik Duvarı ve Ağ Politikaları Uygulama:

- Giden istekleri, yalnızca izin verilen IP adreslerine ve etki alanlarına sınırlamak.

- İç ağdaki hassas sistemlerin dışarıdan erişimini kısıtlamak ve sadece belirli hizmetlere izin vermek.

3. URL Doğrulama ve Filtreleme:

- Kullanıcı tarafından sağlanan URL'leri dikkatlice doğrulamak ve filtrelemek.

- İç IP adreslerine, localhost'a veya özel ağlara yönelik istekleri engellemek.

4. Proxy Kullanımı ve Doğrulama:

- Tüm dış isteklerin belirli bir proxy sunucu üzerinden geçmesini sağlamak ve proxy sunucuda ek güvenlik kontrolleri uygulamak.

- Proxy sunucunun, hedef URL'leri doğrulamasını ve güvenlik politikalarına uygun olup olmadığını kontrol etmesini sağlamak.

5. Girdi Doğrulama ve Sanitizasyon:

- Kullanıcı tarafından sağlanan tüm girdileri dikkatlice doğrulamak ve sanitize etmek.

- URL şemalarını, host isimlerini ve port numaralarını kontrol etmek ve zararlı içerikleri filtrelemek.

6. Detaylı Günlükleme ve İzleme:

- Sunucunun gönderdiği tüm dış istekleri detaylı bir şekilde günlüklemek ve izlemek.

- Anormal veya yetkisiz istekleri tespit etmek ve bu durumlara hızlıca yanıt vermek.

7. Güvenlik Testleri ve Denetimler:

- SSRF açıklarını belirlemek için düzenli olarak güvenlik testleri ve denetimler yapmak.

- Penetrasyon testleri ve otomatik güvenlik tarama araçları kullanarak olası açıkları belirlemek.

Bu yöntemler, API'lerdeki SSRF açıklarını önlemek ve güvenli bir API ortamı oluşturmak için önemlidir.

API8:2023 - Security Misconfiguration

OWASP API Security Top 10 listesinde yer alan ve API'lerin güvenliğini tehlikeye atan yaygın bir güvenlik açığıdır. Güvenlik yapılandırma hataları, API'nin yanlış yapılandırılması veya eksik güvenlik ayarları nedeniyle oluşur. Bu hatalar, saldırganların API'ye veya arka uç sistemlere erişmesine, hassas bilgileri çalmasına veya hizmeti kesintiye uğratmasına yol açabilir.

SECURİTY MİSCONFİGURATİON NEDİR?

- Tanım: API'lerin güvenlik ayarlarının doğru yapılmaması veya eksik bırakılması durumunda ortaya çıkar. Örneğin, gereksiz hizmetlerin etkinleştirilmesi, varsayılan hesapların veya parolaların kullanılması, hatalı hata mesajlarının gösterilmesi gibi durumlar bu kategoriye girer.

- Riskler: Hassas bilgilerin ifşası, yetkisiz erişim, hizmet kesintileri, saldırganların sistemleri ele geçirmesi ve veri bütünlüğünün bozulması.

KORUNMA YÖNTEMLERİ

1. Güvenli Varsayılan Ayarlar Kullanma:

- API ve bağlı hizmetlerin varsayılan olarak güvenli yapılandırmalarla kurulmasını sağlamak.

- Gereksiz hizmetleri devre dışı bırakmak ve kullanılmayan özellikleri kapatmak.

2. Yapılandırma Yönetimi:

- Tüm yapılandırma dosyalarını, ortam değişkenlerini ve gizli anahtarları dikkatlice yönetmek.

- Hassas bilgileri şifreli olarak saklamak ve sadece yetkili kullanıcıların erişimini sağlamak.

3. Düzenli Güvenlik Denetimleri ve Testleri:

- API yapılandırmalarını düzenli olarak denetlemek ve güvenlik açıklarını belirlemek için testler yapmak.

- Otomatik güvenlik tarama araçları ve manuel güvenlik denetimleri kullanarak yapılandırma hatalarını tespit etmek.

4. Detaylı Hata ve Bilgilendirme Mesajları Yönetimi:

- Hata mesajlarını, saldırganların sistem hakkında bilgi edinmesini engelleyecek şekilde yapılandırmak.

- Hata mesajlarında hassas bilgilerin ifşa edilmesini önlemek.

5. Güncelleme ve Yama Yönetimi:

- API yazılımlarını ve bağlı hizmetleri düzenli olarak güncellemek ve yamaları uygulamak.

- Güvenlik açıkları keşfedildiğinde hızlıca güncelleme yapmak ve yamaları dağıtmak.

6. Güvenli Yazılım Geliştirme Yaşam Döngüsü (SDLC):

- Geliştirme sürecinde güvenli kodlama standartlarını ve en iyi uygulamaları takip etmek.

- OWASP ve benzeri güvenlik topluluklarının tavsiyelerini dikkate almak ve güvenlik incelemelerini sürece dahil etmek.

7. Yetkilendirme ve Kimlik Doğrulama Kontrolleri:

- Güçlü kimlik doğrulama ve yetkilendirme kontrolleri uygulamak.

- Kullanıcı erişim seviyelerini dikkatlice yönetmek ve en az yetki ilkesini (Principle of Least Privilege) uygulamak.

8. İzleme ve Olay Müdahale:

- API ve bağlı hizmetlerin sürekli izlenmesini sağlamak.

- Güvenlik olaylarına hızlı bir şekilde yanıt verebilmek için olay müdahale planları oluşturmak ve düzenli olarak test etmek.

Bu yöntemler, API'lerdeki güvenlik yapılandırma hatalarını önlemek ve güvenli bir API ortamı oluşturmak için kritik öneme sahiptir.

API9:2023 - Improper Inventory Management

OWASP API Security Top 10 listesinde yer alan ve API güvenliğini tehdit eden bir diğer önemli güvenlik açığıdır. Bu açık, API'lerin ve bağlı varlıkların (endpoints, versiyonlar, hizmetler, vb.) eksik veya yanlış yönetilmesi durumunda ortaya çıkar. Bu da saldırganların bilinmeyen veya korunmasız API'lere erişmesine, güvenlik açıklarını istismar etmesine yol açabilir.

IMPROPER INVENTORY MANAGEMENT NEDİR?

- Tanım: API'lerin ve bağlı varlıkların doğru bir şekilde envanterinin tutulmaması, güncellenmemesi veya izlenmemesi durumunda meydana gelir. Bu, eski veya güvenliği zayıf API versiyonlarının, belirsiz veya belgelenmemiş endpoints'in varlığına yol açabilir.

- RisklerYetkisiz erişim, veri ihlalleri, hizmet kesintileri, güvenlik açıklarının istismarı ve uyumluluk sorunları.

KORUNMA YÖNTEMLERİ

1. API Envanteri Oluşturma ve Güncelleme:

- Tüm API'lerin, endpoints'lerin, versiyonların ve hizmetlerin detaylı bir envanterini oluşturmak ve sürekli olarak güncellemek.

- API envanterinin, mevcut ve geçmiş API versiyonlarını, kullanılmayan veya eski endpoints'i içermesini sağlamak.

2. Otomatik Envanter Yönetim Araçları Kullanma:

- API envanterini otomatik olarak izleyen ve güncelleyen araçlar kullanmak.

- API gateway'ler, API yönetim platformları ve güvenlik tarama araçları bu süreçte yardımcı olabilir.

3. Düzenli Güvenlik Tarama ve Denetimler:

- API'leri düzenli olarak güvenlik taramaları ve denetimlerden geçirmek.

- Otomatik tarama araçları ve manuel güvenlik denetimleri ile bilinmeyen veya korunmasız API'leri tespit etmek.

4. API Versiyon Yönetimi:

- API versiyonlarını dikkatlice yönetmek ve eski veya desteklenmeyen versiyonları devre dışı bırakmak.

- Yeni API versiyonlarını duyurmak ve kullanıcıları eski versiyonlardan yeni versiyonlara geçmeye teşvik etmek

5. Güvenli Varsayılan Yapılandırmalar:

- API'lerin güvenli varsayılan yapılandırmalarla yayınlanmasını sağlamak.

- Gereksiz veya kullanılmayan endpoints'leri ve özellikleri devre dışı bırakmak.

6. API Belgeleri ve Görünürlük:

- API'lerin, endpoints'lerin ve hizmetlerin detaylı belgelerini oluşturmak ve kullanıcılar için erişilebilir hale getirmek.

- API kullanımını izlemek ve anormal aktiviteleri tespit etmek için loglama ve izleme araçları kullanmak.

7. Güvenlik Politikaları ve Yönergeleri Uygulama:

- API envanter yönetimi için güvenlik politikaları ve yönergeleri oluşturmak.

- Geliştiricilere, sistem yöneticilerine ve güvenlik ekiplerine bu politikaları ve yönergeleri takip etmeleri için eğitim vermek.

8. En Az Yetki İlkesi (Principle of Least Privilege):

- API erişimlerini en az yetki ilkesi ile yönetmek ve yalnızca gerekli erişim izinlerini vermek.

- Yetkisiz erişim denemelerini tespit etmek ve bu durumlara hızlıca yanıt vermek için izleme ve olay müdahale süreçlerini uygulamak.

Bu yöntemler, API'lerdeki hatalı envanter yönetimi açıklarını önlemek ve güvenli bir API ortamı oluşturmak için kritik öneme sahiptir.

API10:2023 - Unsafe Consumption of APIs

OWASP API Security Top 10 listesinde yer alan ve API güvenliğini tehlikeye atan bir diğer önemli güvenlik açığıdır. Bu açık, bir API'nin başka bir API'yi tüketirken güvenlik kontrollerini doğru şekilde uygulamaması durumunda ortaya çıkar. Bu, kötü niyetli API'lerin güvenilir API'ler gibi kullanılmasına ve istemcinin saldırıya maruz kalmasına yol açabilir.

UNSAFE CONSUMPTİON OF APIS NEDİR?

- Tanım:Bir API, başka bir API'yi tüketirken güvenlik kontrollerini ve doğrulamaları ihmal ettiğinde veya yetersiz uyguladığında meydana gelir. Bu, saldırganların, zararlı veya güvenilir olmayan API'leri kullanarak istemciyi kandırmasına ve hassas verilere erişmesine, verileri manipüle etmesine veya hizmeti kesintiye uğratmasına neden olabilir.

- Riskler: Hassas bilgilerin ifşası, veri bütünlüğünün bozulması, hizmet kesintileri, istemcinin güvenilir olmayan kaynaklardan veri alması ve diğer sistemlere yönelik ek saldırılar.

KORUNMA YÖNTEMLERİ

1. Güvenilir API Sağlayıcılarını Kullanma:

- Sadece güvenilir ve doğrulanmış API sağlayıcılarını kullanmak.

- API sağlayıcılarının güvenlik politikalarını, geçmiş güvenlik olaylarını ve güvenilirlik derecelerini gözden geçirmek.

2. API Yanıtlarını Doğrulama:

- Tüketilen API'lerden gelen yanıtları dikkatlice doğrulamak ve filtrelemek.

- Beklenen veri türlerini ve formatlarını doğrulamak ve zararlı veya beklenmeyen verileri reddetmek.

3. Güvenli Bağlantılar Kullanma:

- API iletişimi için güvenli protokoller (örneğin, HTTPS) kullanmak.

- İstemci ve sunucu arasında veri bütünlüğünü ve gizliliğini sağlamak için TLS/SSL kullanmak.

4. Yetkilendirme ve Kimlik Doğrulama Kontrolleri:

- Tüketilen API'ler için güçlü kimlik doğrulama ve yetkilendirme kontrolleri uygulamak.

- API erişim anahtarlarını, OAuth token'larını veya diğer kimlik doğrulama mekanizmalarını güvenli bir şekilde yönetmek ve saklamak.

5. API Gateway ve Güvenlik Ara Katmanı Kullanma:

- API tüketimini yönetmek ve güvenlik kontrollerini uygulamak için API gateway'ler ve ara katmanlar kullanmak.

- API gateway'ler, istekleri izleyebilir, güvenlik politikalarını uygulayabilir ve zararlı trafiği engelleyebilir.

6. Güvenlik Testleri ve Denetimler:

- Tüketilen API'ler için düzenli güvenlik testleri ve denetimler yapmak.

- Otomatik güvenlik tarama araçları ve manuel güvenlik denetimleri kullanarak olası açıkları belirlemek ve düzeltmek.

7. Kullanıcı Girdilerini Doğrulama ve Sanitizasyon:

- API'lere gönderilen tüm kullanıcı girdilerini dikkatlice doğrulamak ve sanitize etmek.

- Zararlı girdilerin API üzerinden diğer sistemlere ulaşmasını engellemek.

8. İzleme ve Olay Müdahale:

- API tüketimini sürekli izlemek ve anormal aktiviteleri tespit etmek.

- Güvenlik olaylarına hızlı bir şekilde yanıt verebilmek için olay müdahale planları oluşturmak ve düzenli olarak test etmek.

Bu yöntemler, API'lerin güvenli olmayan tüketiminden kaynaklanan açıkları önlemek ve güvenli bir API ortamı oluşturmak için kritik öneme sahiptir.