Hackviser Zafiyetler

1-Remote Code Execution

Remote Code Execution (RCE) Zafiyeti Nedir?

Remote Code Execution (Uzak Kod Çalıştırma), bir saldırganın, hedef bir sistemde uzaktan kötü niyetli kod çalıştırmasına olanak tanıyan ciddi bir güvenlik zafiyetidir. Bu tür bir zafiyet, genellikle sistemdeki uygulamaların güvenlik açıklarından, yanlış yapılandırmalardan veya yazılım hatalarından kaynaklanır.

Nasıl Oluşur?

RCE zafiyetleri genellikle aşağıdaki sebeplerden dolayı ortaya çıkar:

1. **Güvensiz Girdi İşleme:**
    - Kullanıcıdan alınan verilerin doğrulanmaması veya kaçış karakterlerinin uygun şekilde işlenmemesi.
    - Örneğin, SQL sorgularında, shell komutlarında veya dosya yükleme işlemlerinde güvenlik kontrolü yapılmaması.
2. **Zayıf Kimlik Doğrulama ve Yetkilendirme:**
    - Sisteme erişim sağlama yetkisinin kötü yapılandırılması.
    - Özellikle admin seviyesindeki işlemlerde giriş doğrulama eksiklikleri.
3. **Eski veya Güncellenmemiş Yazılımlar:**
    - Bilinen zafiyetlere sahip yazılım versiyonlarının kullanılması.
    - Açık kaynak kütüphanelerindeki güncellemelerin ihmal edilmesi.
4. **Yanlış Yapılandırma:**
    - Web sunucuları, veritabanları veya API'lerin yanlış yapılandırılması.
    - Özellikle dosya izinlerinde veya erişim kontrollerinde yapılan hatalar.
5. **Deserialization (Serileştirme) Hataları:**
    - Kötü niyetli verilerin serileştirilmiş bir şekilde sisteme enjekte edilmesi ve sistemde çalıştırılması.

Etkileri Nelerdir?

1. **Sistem Kontrolünün Ele Geçirilmesi:**
    - Saldırgan, hedef sistem üzerinde istediği kodu çalıştırarak tam kontrol sağlayabilir.
2. **Veri İhlalleri:**
    - Hassas kullanıcı verileri (şifreler, kredi kartı bilgileri vb.) çalınabilir.
3. **Hizmet Kesintileri (DoS):**
    - Sunucunun çökertilmesi veya hizmet veremeyecek hale getirilmesi.
4. **Zararlı Yazılım Yükleme:**
    - Fidye yazılımı, virüs veya casus yazılım yüklenebilir.
5. **Ağ Üzerinde Yanal Hareket:**
    - Saldırgan, bir sistemden diğerine geçerek daha geniş çaplı bir saldırı düzenleyebilir.

Kapatılması İçin Öneriler

1. **Güvenli Kodlama Prensiplerini Uygulayın:**

- Kullanıcı girişlerini daima doğrulayın ve sanitize edin.
- Kullanıcıdan gelen verileri doğrudan sistem komutlarına veya SQL sorgularına iletmeyin.

2. **Güncel Yazılım Kullanın:**
   - Uygulamalar, işletim sistemi ve kütüphanelerin güncel sürümlerini kullanın.
   - Bilinen zafiyetleri düzenli olarak takip edin ve yamaları hızlı bir şekilde uygulayın.
3. **Erişim Kontrollerini Güçlendirin:**
   - Kullanıcı rolleri ve izinleri doğru bir şekilde yapılandırın.
   - Admin seviyesindeki işlemlerde iki faktörlü kimlik doğrulama (2FA) gibi ek güvenlik mekanizmalarını kullanın.
4. **Zararlı Kod Çalıştırmayı Engelleyin:**
   - Yazılımınızda serileştirme/deserileştirme işlemlerini güvenli bir şekilde yönetin.
   - Komut çalıştırma veya dosya yükleme işlemlerine erişimi kısıtlayın.
5. **Web Uygulama Güvenlik Duvarı (WAF) Kullanın:**
   - WAF, kötü niyetli trafiği algılayabilir ve engelleyebilir.
6. **Uygulama İzleme ve Loglama:**
   - Sistem aktivitelerini ve giriş/çıkış işlemlerini kaydedin.
   - Anormal davranışları tespit etmek için SIEM araçları kullanın.
7. **Penetrasyon Testleri ve Kod Analizi:**
   - Düzenli olarak uygulamanızın güvenlik testlerini gerçekleştirin.
   - Otomatik kod analizi araçları ile potansiyel güvenlik açıklarını belirleyin.
8. **Eğitim ve Farkındalık:**
   - Geliştirici ekiplerin güvenli yazılım geliştirme konusunda eğitilmesi.

2-Sql İnjection

SQL Injection Nedir?

SQL Injection, saldırganın, web uygulamalarındaki SQL sorgularına kötü niyetli kod enjekte ederek veri çalması, değiştirmesi veya sistem kontrolünü ele geçirmesidir.

Nasıl Oluşur?

1. **Güvensiz Girdi İşleme:** Kullanıcıdan alınan veriler doğrulanmadan SQL sorgularına dahil edilirse.
2. **Parametrik Sorguların Kullanılmaması:** Dinamik sorgularda doğrudan string birleştirme yapılması.
3. **Eski Yazılımlar:** Güvenlik güncellemeleri yapılmamış sistemlerin kullanılması.

Etkileri Nelerdir?

- Veri tabanındaki tüm verilere erişim sağlama.
- Veri çalma veya değiştirme.
- Sistem kontrolünü ele geçirme.

Kapatılması İçin Öneriler

1. **Parametrik Sorgular Kullan:** SQL sorgularında kullanıcı girdisi için *prepared statement* veya *stored procedure* kullanın.
2. **Girdi Doğrulama:** Kullanıcıdan gelen verileri sanitize edin ve kontrol edin.
3. **Güncel Yazılım:** Veri tabanı ve uygulamalarınızı güncel tutun.
4. **Web Uygulama Güvenlik Duvarı (WAF):** Kötü amaçlı trafiği engellemek için kullanın.
5. **Eğitim:** Geliştiricilere güvenli kodlama konusunda eğitim verin.

3- **Server-Side Template Injection (SSTI)**

Server-Side Template Injection (SSTI) Nedir?

SSTI, bir saldırganın, şablon motorlarına (template engines) kötü niyetli veriler enjekte ederek, sunucuda komut çalıştırmasına olanak sağlayan bir güvenlik açığıdır.

Nasıl Oluşur?

1. **Dinamik Şablon İşleme:** Kullanıcı girişlerinin doğrudan şablon koduna dahil edilmesi.
2. **Doğrulama Eksikliği:** Kullanıcı girişlerinin sanitize edilmemesi.
3. **Yanlış Şablon Motoru Kullanımı:** Güvenli olmayan şablon motorlarının veya yanlış yapılandırılmış motorların kullanılması.

Etkileri Nelerdir?

- Sunucuda uzaktan kod çalıştırma (RCE).
- Hassas verilere erişim.
- Sunucunun tamamen ele geçirilmesi.

Kapatılması İçin Öneriler

1. **Girdi Doğrulama:** Kullanıcı girişlerini kontrol edin ve zararlı içerikleri temizleyin.
2. **Şablon Motorlarını Doğru Kullanma:** Güvenli motorlar kullanın ve kullanıcı girdilerini doğrudan dahil etmeyin.
3. **Kod İnceleme:** Şablon entegrasyon kodlarını düzenli olarak gözden geçirin.
4. **Güvenlik Testleri:** SSTI testlerini düzenli olarak uygulayın.

4- Local File Inclusion (LFI)

Local File Inclusion (LFI) Nedir?

Local File Inclusion (LFI), saldırganın, sunucuda bulunan dosyalara yetkisiz erişim sağlamasına ve bunları çalıştırmasına olanak tanıyan bir güvenlik açığıdır. Bu, genellikle dosya yollarının kullanıcı tarafından kontrol edilebilir olduğu durumlarda ortaya çıkar.

Nasıl Oluşur?

1. **Güvensiz Dosya Yolu Girdisi:** Kullanıcıdan alınan dosya adının doğrudan sunucuda bir dosya işleminde kullanılmasından.
2. **Doğrulama Eksikliği:** Dosya yollarının yeterince filtrelenmemesi.
3. **Sunucu Yapılandırma Hataları:** Sunucunun gereksiz dosya izinleri vermesi.

Etkileri Nelerdir?

- **Dosya Erişimi:** Hassas sistem dosyalarını görüntüleme (örneğin, /etc/passwd veya yapılandırma dosyaları).

- **Kod Çalıştırma:** PHP gibi yürütülebilir dosyaların dahil edilmesiyle sistemde komut çalıştırma.
- **Bilgi Sızdırma:** Sistemin dosya yapısı hakkında bilgi edinme.

Kapatılması İçin Öneriler

1. **Girdi Doğrulama ve Kaçış:**
    - Kullanıcıdan gelen dosya girişlerini kısıtlayın ve doğrulayın.
    - Karakterleri temizlemek için realpath() gibi fonksiyonlar kullanın.
2. **Sabit Dosya Yolları Kullanın:**
    - Dosya yollarını sabitleyin ve dinamik kullanıcı girdilerini dahil etmeyin:
3. **İzinleri Kısıtlayın:**
    - Uygulamanın yalnızca gerekli dosya ve dizinlere erişim hakkına sahip olduğundan emin olun.
4. **Sunucu Yapılandırmasını Kontrol Edin:**
    - PHP'de allow_url_include ve allow_url_fopen ayarlarını devre dışı bırakın.
5. **Güvenlik Duvarı ve İzleme:**
    - WAF ile şüpheli LFI girişimlerini engelleyin.

5-XSS

Nedir?

XSS, saldırganların kötü niyetli script'leri bir web uygulamasına enjekte ederek diğer kullanıcıların tarayıcılarında çalıştırmasına olanak tanır.

Nasıl Oluşur?

1. Kullanıcı girdilerinin kaçış yapılmadan HTML veya JavaScript içine dahil edilmesi.
2. Doğrulama ve filtreleme eksikliği.
3. Dinamik içeriklerin güvenliksiz şekilde işlenmesi.

Etkileri

- Kullanıcı oturum bilgilerini çalma.
- Kötü niyetli script'lerin çalıştırılması.
- Kullanıcıyı sahte sayfalara yönlendirme.

Kapatılması İçin Öneriler

1. Girdi Doğrulama: Tüm kullanıcı girişlerini kontrol edin ve zararlı içerikleri temizleyin.
2. Kaçış Yapma: HTML, JavaScript, CSS gibi çıktılarda özel karakterleri kaçış yaparak zararlı kodların çalışmasını engelleyin.
3. Content Security Policy (CSP): Yalnızca izin verilen kaynaklardan gelen script'lerin çalışmasına izin verin.
4. Güvenlik Kitaplıkları Kullanın: OWASP'ın sunduğu araçlar gibi.

6-IDOR

Insecure Direct Object Reference (IDOR) Nedir?

IDOR, bir saldırganın, doğrudan bir nesnenin kimliğini (ID) değiştirerek, yetkisiz erişim sağlamasına olanak tanıyan bir güvenlik açığıdır.

Nasıl Oluşur?

1. Kimlik Doğrulama veya Yetkilendirme Eksikliği:
   Kullanıcı, yalnızca yetkili olduğu kaynaklara erişimle sınırlandırılmamışsa.

2. Doğrudan Nesne Referansı:
   Örneğin, bir URL'de kullanıcıya ait bir dosya kimliğinin değiştirilmesiyle erişim:

Etkileri

- Yetkisiz verilere erişim (ör. başka bir kullanıcının hesap bilgileri).
- Hassas veri ihlali.
- Sisteme zarar verme veya kötüye kullanım.

Kapatılması İçin Öneriler

1. Yetkilendirme Kontrolü:
   Her kullanıcı isteğinde, erişim haklarının doğrulanmasını sağlayın.

2. Nesne Referanslarını Rastgeleleştirme:
   Doğrudan ID yerine, rastgele veya şifrelenmiş kimlikler kullanın.

3. Loglama ve İzleme:
   Yetkisiz erişim denemelerini tespit etmek için sistem loglarını düzenli olarak inceleyin.

7-CSRF

CSRF Nedir?

Cross-Site Request Forgery (CSRF), saldırganların, bir kullanıcının kimlik doğrulamasıyla yapılmış istekleri, kullanıcının bilgisi olmadan ve izni dışında, hedef bir web sitesine göndermesine olanak tanıyan bir güvenlik açığıdır. Bu saldırı türü, kullanıcının tarayıcısındaki mevcut oturum bilgilerini kullanarak yapılır.

Nasıl Oluşur?

1. Kimlik Doğrulaması:
   Kullanıcı, bir web sitesine giriş yapmış ve oturum açmıştır.

2. Sahte İstek Gönderme:
   Saldırgan, kullanıcının tarayıcısına kötü niyetli bir istek göndermek için bir link veya form sunar. Kullanıcı bu istekleri fark etmeden gönderir.

Etkileri:

- Yetkisiz İşlemler:
  Kullanıcının hesabıyla işlem yapılabilir, örneğin para transferi veya şifre değiştirme.
- Hesap Ele Geçirme:
  Şifre sıfırlama işlemleriyle kullanıcının hesabı ele geçirilebilir.
- Veri İhlali:
  Kullanıcının kişisel verileri sızdırılabilir.

Kapatılması İçin Öneriler:

1. CSRF Token Kullanımı:
   Her formda benzersiz bir token kullanın ve bunu doğrulayın. Token'lar yalnızca geçerli oturumlar için geçerlidir.

2. Referer Kontrolü:
   İsteklerin, güvenilir kaynaklardan geldiğinden emin olun.

3. SameSite Çerez Politikası:
   Çerezlerde SameSite özelliğini etkinleştirerek üçüncü taraf sitelerden gelen istekleri engelleyin.

4. GET ve POST Ayrımı:
   Veritabanı ve veri değiştiren işlemler için sadece POST isteklerini kullanın, GET işlemleri yalnızca veri alımı için kullanılmalı.

8- Broken Authentication

Broken Authentication (Bozuk Kimlik Doğrulama) Nedir?

Broken Authentication, bir web uygulamasında kimlik doğrulama mekanizmalarının zayıf olması veya hatalı yapılandırılması sonucu, saldırganların kullanıcı hesaplarına yetkisiz erişim sağlamasına imkan tanıyan bir güvenlik açığıdır. Bu, kimlik doğrulama sürecinde bulunan zafiyetlerden kaynaklanabilir ve genellikle kötü şifreleme, hatalı şifre sıfırlama işlemleri veya kullanıcı bilgilerini düzgün yönetmeme gibi sorunlarla ortaya çıkar.

Nasıl Oluşur?

1. Zayıf Şifre Politikaları:
   Kullanıcılar, kolay tahmin edilebilen veya sık kullanılan şifreleri tercih ederler.

2. Şifrelerin Yanlış Saklanması:
   Şifrelerin düz metin olarak depolanması ya da zayıf şifreleme algoritmalarının kullanılması.

3. Oturum Yönetimi Hataları:
   Oturum çerezlerinin düzgün bir şekilde güvence altına alınmaması, çerezlerin kolayca ele geçirilmesi veya oturum sonlandırma işlemlerinin hatalı yapılması.

4. Çok Faktörlü Kimlik Doğrulama Eksikliği (MFA):
   Kullanıcı kimlik doğrulamasında yalnızca şifre kullanmak, ek güvenlik katmanlarının (örneğin SMS veya uygulama tabanlı doğrulama) eksik olması.

5. Şifre Sıfırlama ve Oturum Zafiyetleri:
   Şifre sıfırlama süreçlerinin yeterince güvenli olmaması, saldırganların başkasının hesabını ele geçirmesine yol açabilir.

Etkileri:

- Yetkisiz Erişim:
  Kullanıcıların hesaplarına, bankacılık hesapları, e-posta gibi hassas bilgilere yetkisiz erişim sağlanabilir.

- Hesap Ele Geçirme:
  Saldırganlar, başkalarının hesaplarına erişip, önemli işlemler yapabilir, veri çalabilir veya kötüye kullanabilir.

- Kimlik Bilgisi Hırsızlığı:
  Kullanıcıların kimlik bilgileri çalınabilir ve bu bilgiler başka saldırılar için kullanılabilir.

Kapatılması İçin Öneriler:

1. Şifre Güvenliği:
   - Güçlü şifre politikaları belirleyin (en az 12 karakter, büyük ve küçük harfler, rakamlar ve özel karakterler içersin).
   - Şifrelerin hash edilerek saklanması için güçlü algoritmalar (örneğin bcrypt, scrypt veya Argon2) kullanın.

2. Çok Faktörlü Kimlik Doğrulama (MFA):

- Kullanıcıların kimlik doğrulamasını güçlendirmek için MFA kullanın (SMS, e-posta doğrulama, uygulama tabanlı doğrulama vb.).

3. Oturum Yönetimi:
   - Çerezleri güvenli bir şekilde yönetin: Çerezlerin HttpOnly, Secure ve SameSite parametrelerini kullanarak korunmasını sağlayın.
   - Oturum süresi kısıtlamaları ve oturum sonlandırma: Uzun süreli oturumların sonlandırılması ve oturum çerezlerinin düzgün şekilde sıfırlanması.
4. Şifre Sıfırlama Güvenliği:
   - Şifre sıfırlama işlemi için ekstra güvenlik kontrolleri ekleyin (örneğin, güvenlik soruları, e-posta doğrulama).
   - Şifre sıfırlama bağlantılarının zaman sınırlı ve tek kullanımlık olmasını sağlayın.
5. Güvenli Kimlik Doğrulama Akışları:
   - Oturum açma süreçlerinin doğru şekilde yapılandırıldığından ve hatalı kimlik doğrulama işlemlerinin düzgün işlediğinden emin olun. Örneğin, doğru şifre girilmediğinde deneme limitleri ve hesap kilitleme uygulanabilir.