Yavuzlar OWASP TOP 10 Ödev

OWASP TOP 10

1-) Broken Accsess Control

Broken Accsess Control nedir ?

Broken Accsess Control, yetkisiz kullanıcıların erişmemesi gereken verilere erişmesine, bunları değiştirmesine veya silmesine izin veren bir güvenlik açığıdır. Bu güvenlik açığı, en kritik web uygulaması güvenliklerinden biri olarak kabul edilir

Örnek olarak açıklamak gerekirse;

Hedef web sitesinde kullanıcının erişim yetkileri dışında yanlış bir şekilde yönettiği bir durum. Örnek olarak web sitesinde bir yere login oldun ve oradan geri çıkış yaptın çıkış yaptığın zaman senin normal şartlarda sessions yani oturumun kapanmalı kapanmazsa bu açık var demektir geri tuşuna bastığında seni tekrardan bi sessions yani oturum içerisine alır. Bi başka örnek olarak ise web sitenin kaynak kodlarında yorum satırı içerisinde bir bilgi unutulması, urlde kullanıcının idsi yazması gibi durumlarda bu açığa örnek gösterilebilir.

Bir web uygulaması, kullanıcıların belirli bir URL'ye erişmesine izin verir, ancak bu URL'nin erişim kontrolü düzgün yapılmamıştır. Örneğin:

- Kullanıcı 1, `/user/1/profile` URL'sine giderek kendi profilini görebilir.

Bir web uygulamasında, sadece yöneticilerin erişebilmesi gereken bir yönetim paneli vardır:

- Yönetici `/admin/dashboard` URL'sine giderek yönetim paneline erişebilir.

Nasıl Önlenir ?

1. Sürekli Denetim ve Test Erişim Kontrolü:

Erişim kontrol mekanizmasının etkin bir şekilde sürekli test edilmesi ve denetlenmesi

2. Varsayılan Olarak Erişimi Reddet:

Herkesin kaynaklara ve işlevlere erişemeyeceği şekilde erişim kontrolü tasarlanması

4. Rol Tabanlı Erişim Kontrolünü Etkinleştirilmesi:

Bu, yaygın olarak kullanılan bir erişim kontrol mekanizmasıdır. Buna göre, kullanıcılara rollerine göre izinler verilir. Her kullanıcıyı ayrı ayrı tanımlamak yerine, kullanıcılar bir rol grubuna atanır, bu şekilde BT

desteği ve yönetiminin mücadelesi azaltılabilir ve operasyonel verimlilik en üst düzeye çıkarılır.

5. İzin Tabanlı Erişim Kontrolünü Etkinleştirilmesi:

Bu, yetkilendirme katmanının kullanıcının belirli verilere erişim veya belirli bir eylemi gerçekleştirme iznine sahip olup olmadığını, genellikle kullanıcının rollerinin bu izne sahip olup olmadığını kontrol ederek kontrol ettiği bir erişim kontrol yöntemidir.

6. Zorunlu erişim kontrolünü etkinleştirilmesi:

Bu, kaynağın içerdiği bilginin hassasiyetine göre kaynaklara erişim yeteneğini kısıtlayan bir güvenlik yöntemidir. Bu güvenlik politikası yalnızca yönetici tarafından kontrol edilebilir, normal kullanıcılar bu politikayı değiştirme yetkisine sahip değildir.

2-) Cryptographic Failures

Cryptographic Failures Nedir ?

Normal şartlarda şifreli halde bulunması gereken data veya bilimum diğer şeylerin olması gerektiği gibi olmayıp salt, çözümlü bir biçimde tutulması veya aktarılması durumu. Bunu söyle örnekleyelim... Bir web sitesinde https protokolü ile iletişim sağlanıyor olması gerekirken http olarak sağlanması bu sorunu meydana getirir. Bununla birlikte bazı doğrulama kodları ve şifreler gibi normal şartlar altında şifreli bir şekilde güvenle tutulması gereken şeylerin yine plaintext halde tutulması durumu.

Kriptografik hatalara yol açan bazı durumlar:

Verilerin şifrelenmemiş olarak iletilmesi

Zayıf güvenlik algoritmalarının kullanılması

Key’lerin korunamaması

Kriptografik algoritmalardaki fonksiyonların yeterince rastgele değerler üretememesi

Padding yani verinin nerede başlayıp nerede bittiğini zorlaştırmak için eklenen verilerin eklenmesi sırasında eski metodların kullanılması

Dijital sertifikaların düzgün şekillerde doğrulanamaması

Örnek olarak

Zayıf Şifreleme Algoritmalarının Kullanımı

Eski teknoloji bir kasa düşün, açılması çok kolay. MD5 ya da SHA-1 gibi zayıf şifreleme algoritmalarını kullanmak da işte böyle bir şey. Saldırganlar, şifrelenmiş veriyi bu algoritmalarla kolayca çözebilir. Günümüzde, artık bu algoritmalardan uzak durmak gerekiyor.

Bunları önlemek için:

Saklanan veriler sınıflandırılmalı

Hassas olarak sınıflandırılmış gereksiz veriler saklanmamalı

Gerekli hassas veriler şifrelenerek saklanmalı

Güçlü şifreleme algoritmaları kullanılmalı

Anahtarların güvenliği sağlanmalı

Sıradan şifreleme algoritmaları yerine özgünlüğü de kanıtlayan Galois Counter Mode gibi algoritmalar kullanılmalı

3-) İnjection

İnjection (Enjeksiyon) Nedir?

Enjeksiyon saldırıları, saldırganların bir uygulamada bulunan güvenlik açıklarını istismar ederek sisteme kötü amaçlı kod göndermesiyle gerçekleşir. Bu tür bir istismar, saldırganların yetkisiz komutlar yürütmesine, verilere erişmesine veya sistemin işlemlerini manipüle etmesine olanak tanıyabilir. Enjeksiyon saldırıları çok tehlikeli olabildiğinden ve yaygın olarak kullanıldığından, günümüzde siber güvenlik için kritik bir tehdit oluştururlar.

Türleri Neler ?

- **SQL injection**
- **Command injection**
- **XML injection**
- **HTML injection**

SQL İnjection Nedir?

SQL enjeksiyonu, SQLI olarak da bilinir, kötü amaçlı SQL kodunu arka uç veritabanı manipülasyonu için kullanarak görüntülenmesi amaçlanmayan bilgilere erişmek için kullanılan yaygın bir saldırı vektörüdür. Bu bilgiler, hassas şirket verileri, kullanıcı listeleri veya özel müşteri ayrıntıları dahil olmak üzere herhangi bir sayıda öğeyi içerebilir.

SQL enjeksiyonunun bir işletme üzerindeki etkisi çok geniş kapsamlıdır. Başarılı bir saldırı, kullanıcı listelerinin yetkisiz görüntülenmesi, tüm tabloların silinmesi ve bazı durumlarda saldırganın bir veritabanına yönetici hakları elde etmesiyle sonuçlanabilir; bunların hepsi bir işletme için oldukça zararlıdır.

Bir SQLi'nin potansiyel maliyetini hesaplamak, telefon numaraları, adresler ve kredi kartı bilgileri gibi kişisel bilgilerin çalınması durumunda müşteri güveninin kaybolmasını dikkate almak önemlidir.

Bu vektör herhangi bir SQL veritabanına saldırmak için kullanılabilirken, en sık hedefler web siteleridir.

SQL Enjeksiyonlarının Türleri

SQL enjeksiyonları genellikle üç kategoriye ayrılır:

- **Bant içi SQLi (Klasik)**
- **Çıkarımsal SQLi (Kör)**
- **Bant dışı SQLi.**
- **İkinci dereceden SQL enjeksiyonu**
- **Zaman tabanlı SQL enjeksiyonu**
- **Union tabanlı SQL enjeksiyonu**
- **Hata tabanlı SQL enjeksiyonu**

SQL enjeksiyonlarının türlerini, arka uç verilerine erişmek için kullandıkları yöntemlere ve hasar potansiyellerine göre sınıflandırabilirsiniz.

Bant içi SQLi

Saldırgan saldırılarını başlatmak ve sonuçlarını toplamak için aynı iletişim kanalını kullanır. Bant içi SQLi'nin basitliği ve verimliliği onu en yaygın SQLi saldırı türlerinden biri yapar. Bu yöntemin iki alt çeşidi vardır:

- **Hata tabanlı SQLi** — saldırgan, veritabanının hata mesajları üretmesine neden olan eylemler gerçekleştirir. Saldırgan, bu hata mesajlarının sağladığı verileri, veritabanının yapısı hakkında bilgi toplamak için kullanabilir.
- **Birlik tabanlı SQLi** —bu teknik, veritabanı tarafından oluşturulan birden fazla seçme ifadesini birleştirerek tek bir HTTP yanıtı elde eden UNION SQL operatöründen yararlanır. Bu yanıt, saldırgan tarafından yararlanılabilecek veriler içerebilir.

Çıkarımsal (Kör) SQLi

Saldırgan sunucuya veri yükleri gönderir ve yapısı hakkında daha fazla bilgi edinmek için sunucunun yanıtını ve davranışını gözlemler. Bu yönteme kör SQLi denir çünkü veriler web sitesi veritabanından saldırgana aktarılmaz, böylece saldırgan bant içi saldırı hakkında bilgi göremez.

Kör SQL enjeksiyonları sunucunun tepki ve davranış kalıplarına güvenir, bu nedenle genellikle yürütülmesi daha yavaştır ancak aynı derecede zararlı olabilir. Kör SQL enjeksiyonları aşağıdaki gibi sınıflandırılabilir:

- **Boolean** — saldırganın veritabanına bir SQL sorgusu göndererek uygulamanın bir sonuç döndürmesini istemesi. Sonuç, sorgunun doğru veya yanlış olmasına bağlı olarak değişecektir. Sonuca bağlı olarak, HTTP yanıtındaki bilgiler değişecek veya değişmeden kalacaktır. Saldırgan daha sonra mesajın doğru veya yanlış bir sonuç üretip üretmediğini anlayabilir.
- **Zaman tabanlı** — saldırgan veritabanına bir SQL sorgusu gönderir, bu da veritabanının tepki vermeden önce beklemesini (saniye cinsinden bir süre) sağlar. Saldırgan, veritabanının yanıt vermesinin aldığı zamandan, bir sorgunun doğru mu yoksa yanlış mı olduğunu görebilir. Sonuca göre, bir HTTP yanıtı anında veya bir bekleme süresinin ardından oluşturulur. Saldırgan böylece veritabanındaki verilere güvenmeden, kullandıkları mesajın doğru mu yoksa yanlış mı döndürdüğünü anlayabilir.

Bant dışı SQLi

Saldırgan bu saldırı biçimini yalnızca web uygulaması tarafından kullanılan veritabanı sunucusunda belirli özellikler etkinleştirildiğinde gerçekleştirebilir. Bu saldırı biçimi öncelikle bant içi ve çıkarımsal SQLi tekniklerine bir alternatif olarak kullanılır.

Bant dışı SQLi, saldırganın saldırıyı başlatmak ve bilgi toplamak için aynı kanalı kullanamadığı veya bir sunucunun bu eylemlerin gerçekleştirilmesi için çok yavaş veya dengesiz olduğu durumlarda gerçekleştirilir. Bu teknikler, sunucunun bir saldırgana veri aktarmak için DNS veya HTTP istekleri oluşturma kapasitesine güvenir.

İkinci dereceden SQL enjeksiyonu

Web uygulamalarında kullanıcı tarafından sağlanan verilerin bir veri tabanında saklandığı ve saklanan veriler daha sonra farklı bir bağlamda kullanıldığında kötü amaçlı SQL kodunun çalıştırıldığı bir güvenlik açığı türüdür. Bu durum, uygulama depolanan verileri düzgün bir şekilde sterilize etmediğinde veya doğrulamadığında ortaya çıkabilir.

Zaman tabanlı SQL enjeksiyonu

Zaman tabanlı saldırılar, enjekte edilen SQL kodunun çalışıp çalışmadığını belirlemek için veri tabanının yanıtındaki gecikme veya zamanlama farklılıklarından yararlanır. Bilgisayar korsanı, zaman gecikmeli SQL deyimleri enjekte ederek, uygulamanın yanıt vermesi için geçen süreyi gözlemleyerek bilgi çıkarabilir veya veri tabanını manipüle edebilir.

Union tabanlı SQL enjeksiyonu

Union tabanlı saldırılar, birden fazla SELECT deyiminin sonuç kümelerini birleştirmek için SQL'deki UNION operatöründen yararlanır. Bilgisayar korsanı, hazırlanmış bir UNION deyimi enjekte ederek diğer veri tabanı tablolarından veri alabilir veya açığa çıkması amaçlanmayan ek bilgileri çıkarabilir.

Hata tabanlı SQL enjeksiyonu

Hata tabanlı saldırılar, bilgi elde etmek için veri tabanı tarafından oluşturulan hata mesajlarından ve yanıtlarından yararlanır. Bilgisayar korsanı, bir hatayı tetikleyen kötü amaçlı SQL kodu enjekte ederek veri tabanı yapısı, tablo adları veya belirli veriler hakkında ayrıntılar elde edebilir.

SQLI önleme ve azaltma

- Yasa dışı kullanıcı girdilerini tespit edebilen kod yazma uygulaması olan girdi doğrulama
- SQLI'yi ve diğer çevrimiçi tehditleri filtrelemek için web uygulama güvenlik duvarı (WAF)
- Parametrelendirilmiş sorgular içeren hazır deyimler
- Girdi doğrulama
- En az ayrıcalık ilkesi
- Saklı yordamlar (bir veri tabanında tanımlanıp saklanan ve daha sonra bir uygulamadan çağrılabilen SQL)

Command injection (Komut Enjeksiyonu) Nedir?

Komut enjeksiyonu, savunmasız bir uygulama aracılığıyla ana işletim sisteminde keyfi komutların yürütülmesini amaçlayan bir saldırı türüdür. Bu tür saldırılar, uygulamanın güvenli olmayan kullanıcı tarafından sağlanan verileri (örneğin, formlar, çerezler, HTTP başlıkları vb.) sistem kabuğuna iletmesi durumunda gerçekleşebilir. Saldırgan tarafından sağlanan işletim sistemi komutları genellikle uygulamanın ayrıcalıklarıyla çalıştırılır. Komut enjeksiyonu saldırıları genellikle yetersiz giriş doğrulamasından kaynaklanır.

Bu saldırı, Kod Enjeksiyonundan farklıdır; çünkü kod enjeksiyonu saldırganın kendi kodunu eklemesine ve uygulamanın bu kodu çalıştırmasına izin verirken, Komut Enjeksiyonunda saldırgan, uygulamanın mevcut işlevselliğini sistem komutlarıyla genişletir, kendi kodunu eklemeye gerek kalmadan.

Neden Kaynaklanır ?

- **Yetersiz Giriş Doğrulaması**: Kullanıcıdan alınan veriler üzerinde yeterli doğrulama yapılmaması, zararlı komutların sisteme iletilmesine izin verebilir.
- **Kötü Kodlama Uygulamaları**: Uygulama geliştiricilerinin, kullanıcı girişlerini doğrudan sistem komutlarına iletmesi veya bu girişleri doğru bir şekilde kaçırmadan kullanması durumunda komut enjeksiyonu meydana gelebilir.
- **Veri Temizleme Eksiklikleri**: Kullanıcıdan alınan verilerin uygun şekilde temizlenmemesi veya filtrelenmemesi, zararlı komutların sisteme enjekte edilmesine yol açabilir.
- **Güvenlik Açıkları**: Uygulamanın sistem komutlarını yürütme yeteneği, bu komutlara yetkisiz erişim sağlamaya yönelik açıklar içerebilir.

- **Yanlış Yapılandırılmış Sistemler**: Sistem veya uygulama yapılandırmalarındaki yanlışlıklar, komut enjeksiyonuna karşı korumasız hale gelmesine neden olabilir.

Türleri

- **Sistem Komut Enjeksiyonu**: En yaygın komut enjeksiyonu türüdür. Saldırgan, uygulamanın sistem komutlarını çalıştırmasına izin veren bir güvenlik açığını kullanarak komutlar enjekte eder. Örneğin, `ls`, `cat`, `rm` gibi komutları içerebilir.
- **Shell Komut Enjeksiyonu**: Bu türde, saldırgan komutları bir shell (kabuk) aracılığıyla çalıştırır. Bu, genellikle uygulamanın komutları doğrudan bir shell'e gönderdiği durumlarda görülür. Komutların kabuk (shell) komutlarıyla işlenmesi, enjeksiyon için bir fırsat sağlar.
- **SQL Komut Enjeksiyonu**: Veritabanı komutlarına odaklanır. Saldırgan, SQL sorgularına zararlı kodlar ekler ve veritabanına yetkisiz erişim sağlar veya verileri manipüle eder. SQL enjeksiyonu, veritabanı üzerinde tam kontrol elde etme potansiyeline sahiptir.
- **OS Komut Enjeksiyonu**: İşletim sistemi komutlarının enjekte edilmesini içerir. Bu türde, uygulama işletim sistemi komutlarını yürütme yeteneğine sahipse, saldırgan bu komutları manipüle edebilir.
- **XPath Enjeksiyonu**: XML verileriyle çalışan uygulamalarda, XPath sorgularına zararlı veriler ekleyerek XML veritabanlarına yetkisiz erişim sağlar.
- **LDAP Enjeksiyonu**: LDAP (Lightweight Directory Access Protocol) sorgularına zararlı veriler enjekte eder. Bu tür enjeksiyon, dizin hizmetlerine yetkisiz erişim veya manipülasyon sağlamaya yönelik olabilir.

Nasıl engellenir ?

- Giriş Doğrulama ve Filtreleme
- Hazırlanmış İfadeler ve Parametreli Sorgular Kullanma
- Güvenlik Testleri ve Kod İncelemesi
- Minimum Ayrıcalık İlkesini Uygulama
- Güvenli Programlama Uygulamaları
- Web Güvenlik Duvarları ve Filtreleme
- Yazılım Güncellemeleri ve Yamalar

XML İnjection (XML Enjeksiyonu) Nedir ?

XML enjeksiyonu, bir saldırganın XML verilerini işleyen bir uygulamaya kötü niyetli XML içeriği enjekte ettiği bir saldırı türüdür. Bu saldırı, XML'i veri alışverişi veya yapılandırma için kullanan uygulamalardaki güvenlik açıklarını hedef alır, örneğin web hizmetleri veya yapılandırma dosyaları gibi.

XML Enjeksiyonunun Nasıl Çalıştığı

1. **Kötü Niyetli XML Enjeksiyonu**: Saldırgan, özel olarak hazırlanmış XML verilerini kötü niyetli yüklerle gönderir. Bu yükler, XML ayrıştırmasını değiştirmek veya beklenmeyen eylemler

gerçekleştirmek üzere tasarlanabilir.

2. **XML Verisinin Manipülasyonu**: Enjekte edilen XML, XML belgesinin beklenen yapısını veya içeriğini değiştirebilir. Bu, beklenmeyen davranışlara neden olabilir veya hassas verilerin açığa çıkmasına yol açabilir.
3. **XML Ayrıştırıcılarının Sömürülmesi**: Birçok uygulama XML verilerini okumak ve işlemek için XML ayrıştırıcıları kullanır. Uygulama, verileri doğru şekilde doğrulamaz veya temizlemezse, enjekte edilen XML ayrıştırıcı güvenlik açıklarını sömürerek güvenlik sorunlarına yol açabilir.

XML Enjeksiyonu Türleri

1. **XML Varlık Enjeksiyonu**: Kötü niyetli XML varlıklarını veri içerisine enjekte eder. XML ayrıştırıcıları bu varlıkları genişlettiğinde, bilgi sızdırma veya Hizmet Reddi (DoS) saldırılarına neden olabilir. Örneğin:
   `<!ENTITY xxe SYSTEM "file:///etc/passwd"> <data>&xxe;</data>`
2. **XPath Enjeksiyonu**: Saldırgan, XML verilerine kötü niyetli XPath sorguları enjekte eder. Bu, XPath ifadesi tarafından döndürülen sonuçları manipüle edebilir ve yetkisiz veri erişimine veya uygulama mantığının değiştirilmesine neden olabilir.
3. **XML Dış Varlık (XXE) Enjeksiyonu**: XML ayrıştırıcılarının dış varlıkları dahil etmesine izin vermesini hedefler. Dış varlık referanslarını enjekte ederek, bir saldırgan yerel dosyalara erişebilir veya diğer yetkisiz eylemleri gerçekleştirebilir. Örneğin:
   `<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]> <foo>&xxe;</foo>`
4. **XML Bombası**: XML bombası, ayrıştırıcının aşırı kaynak tüketmesine neden olabilecek özel olarak hazırlanmış bir XML belgesidir. Bu, uygulamanın çökmesine veya yanıt vermemeye neden olabilir. Örneğin:
   `<foo> <bar>&#60;!DOCTYPE foo [<!ELEMENT foo (#PCDATA)><!ENTITY xxe SYSTEM "file:///etc/passwd">]&#62;</bar> </foo>`

Önleme Yöntemleri

- **Giriş Doğrulama ve Temizleme**: XML verilerini işlemeye başlamadan önce, kullanıcıdan alınan tüm girişler doğrulanmalı ve temizlenmelidir. Kötü niyetli içerik veya beklenmeyen yapılar içermediğinden emin olunmalıdır.
- **XML Dış Varlıkları Devre Dışı Bırakma**: XML ayrıştırıcılarını, dış varlıkların işlenmesini engelleyecek şekilde yapılandırılmalıdır. Bu, XXE saldırılarına karşı koruma sağlar.
- **Güvenli XML Kütüphaneleri Kullanma**: Güvenlik açıklarını minimize eden ve XML verilerini güvenli bir şekilde işlemek için tasarlanmış XML kütüphaneleri ve ayrıştırıcıları kullanılmalıdır.
- **Veri Erişimini Sınırlama**: Hassas verilere erişimi sınırlandırılmalı ve uygulamanın işlediği veya döndürdüğü verilerin uygun şekilde korunduğundan emin olunmalıdır.
- **Güvenlik En İyi Uygulamalarını Uygulama**: XML verilerini işlemek ve yönetmek için güvenlik en iyi uygulamalarına uyulmalı ve düzenli olarak güvenlik denetimleri gerçekleştirilmelidir.

HTML İnjection (HTML ENJEKSİYONU) Nedir?

HTML Enjeksiyonu, Cross Site Scripting olarak da bilinir. Bir saldırganın diğer kullanıcılar tarafından görüntülenen web sayfalarına HTML kodu enjekte etmesine olanak tanıyan bir güvenlik açığıdır.

Saldırganlar genellikle kullanıcıyı aldatmak ve onlardan veri toplamak için savunmasız uygulamalara kötü amaçlı JavaScript, VBScript, ActiveX ve/veya HTML enjekte eder. Cross-site scripting (XSS) güvenlik açıkları saldırganlar tarafından sisteminizdeki hassas verilere erişerek oradaki kimlik doğrulama kontrollerini atlatmak için kullanılabilir. İyi hazırlanmış kötü amaçlı kod, saldırganın tüm sisteme erişmesine bile yardımcı olabilir.

Html Enjeksiyonunun Örneği

- Saldırgan ilk olarak HTML enjeksiyonuna karşı savunmasız bir site bulur
- Daha sonra saldırgan, URL'ye enjekte edilen kötü amaçlı kodu e-posta veya başka bir mekanizma aracılığıyla kurban kullanıcıya gönderir.
- Mağdur kullanıcı bu kötü amaçlı URL'ye tıkladığında, kurban kullanıcının ayrıcalıklarına sahip JavaScript veya VBScript kodu çalıştırılır.
- Yürütülen koda bağlı olarak kullanıcının hassas bilgilerini ifşa edebilir veya hatta kurbanın bilgisayarını tehlikeye atabilir.

Neden kaynaklanır?

- **Yetersiz Giriş Doğrulama ve Temizleme**
- **Eksik Çıktı Kodlama**
- **Yetersiz İçerik Güvenliği**
- **Dinamik İçerik Oluşturma**
- **Zayıf Kullanıcı Doğrulama**
- **Güvenlik Açıkları ve Hatalar**

Nasıl Önlenir ?

- **Giriş Doğrulama ve Temizleme**: Kullanıcıdan alınan tüm giriş verilerini sıkı bir şekilde doğrulanabilir ve zararlı karakterleri temizlenebilir.
- **Çıktı Kodlama**: Kullanıcıdan alınan verileri HTML sayfalarına yerleştirirken doğru şekilde kodlanmalı. Örneğin, özel karakterleri HTML entity kodlarıyla değiştirilmeli.
- **İçerik Güvenliği Politikaları**: İçerik güvenliği politikalarını uygulayarak, kötü niyetli kodların sayfalarda çalışması engellenebilir.
- **Güvenli Yazılım Geliştirme**: Güvenli kodlama pratiklerine uyarak ve düzenli güvenlik denetimleri yaparak, HTML enjeksiyonuna karşı koruma sağlanılabilir.

4-) Insecure Design (Güvensiz Tasarım)

Güvensiz Tasarım Nedir?

OWASP Top 10'da sıklıkla **A04** olarak anılan **güvensiz tasarım , bir uygulamanın mimarisinde ve güvenlik kontrollerinde temel bir kusuru** ifade eder . Belirli güvenlik açıklarına değil, bunları mümkün kılan temel nedene atıfta bulunur. Bunu, bir binayı çeşitli çatlaklara ve hasara karşı hassas hale getiren zayıf bir temel gibi düşünebiliriz.

Neden kaynaklanır ?

- **Güvenlik İlkelerinin İhmal Edilmesi**: Tasarım sürecinde güvenlik standartlarının ve en iyi uygulamalarının dikkate alınmaması.
- **Yetersiz Tehdit Modelleme**: Potansiyel saldırı vektörlerinin ve tehditlerin yeterince analiz edilmemesi.
- **Güvenlik Özelliklerinin Eksikliği**: Şifreleme, kimlik doğrulama ve yetkilendirme gibi temel güvenlik özelliklerinin tasarıma dahil edilmemesi.
- **Yanlış Varsayımlar**: Sistemin güvenliği ile ilgili yanlış varsayımlar yapmak, örneğin, tüm kullanıcıların daima güvenilir olduğu varsayımı.
- **Güncellemeler ve Bakım Eksikliği**: Güvenlik açıkları ve zayıflıkları düzeltecek şekilde düzenli güncellemeler ve bakım yapılmaması.

Türleri

- **Yetersiz Kimlik Doğrulama ve Yetkilendirme**: Kullanıcıların kimliklerini doğrulamak ve yetkilerini belirlemek için yeterli mekanizmaların olmaması. Örneğin, zayıf şifreleme, kolay tahmin edilebilir şifreler veya eksik iki faktörlü kimlik doğrulama.
- **Veri Koruma Eksiklikleri**: Verilerin yeterince korunmaması veya şifrelenmemesi. Bu, hassas bilgilerin yetkisiz kişiler tarafından erişilmesine veya ele geçirilmesine yol açabilir.
- **Güvenlik Açıkları İçin Test Edilmeyen Bileşenler**: Yazılımda kullanılan üçüncü taraf bileşenlerin veya kütüphanelerin güvenlik açıklarının değerlendirilmemesi.
- **Hatalı Giriş Doğrulama**: Kullanıcı girişlerinin yeterince doğrulanmaması, bu da SQL enjeksiyonu, XSS (Cross-Site Scripting) gibi saldırılara açık olabilen güvenlik açıklarına neden olabilir.
- **Güvenlik Özelliklerinin Eksikliği**: Güvenlik duvarları, izleme sistemleri ve diğer koruyucu özelliklerin eksik olması.
- **Yanlış Yapılandırma**: Sistem veya uygulama bileşenlerinin güvenlik açısından yanlış yapılandırılması. Bu, varsayılan şifreler veya izinlerin yanlış ayarlanmış olması gibi sorunlara yol açabilir.

Nasıl Önlenir ?

- **Güvenli Tasarım İlkeleri Benimseme**: Tasarım aşamasında güvenlik prensiplerini dikkate almak.
- **Tehdit Modelleme**: Sistem tasarımında potansiyel tehditleri ve saldırı vektörlerini analiz etmek.

- **Güvenlik Özelliklerinin Entegre Edilmesi**: Şifreleme, kimlik doğrulama, yetkilendirme ve veri bütünlüğü gibi güvenlik özelliklerini tasarım aşamasında entegre etmek.
- **Güvenlik Testleri ve Denetimler**: Tasarım ve geliştirme sürecinde güvenlik testleri ve denetimleri gerçekleştirmek.
- **Eğitim ve Farkındalık**: Yazılım geliştirme ekibini güvenlik konularında eğitin ve güvenlik farkındalığını artırmak.
- **Güvenli Kodlama Pratikleri**: Güvenli kodlama standartlarını ve en iyi uygulamaları takip etmek.
- **Güncellemeler ve Bakım**: Yazılım sistemlerinin düzenli olarak güncellenmesi ve bakımı.
- **Güvenlik Gereksinimlerinin Belirlenmesi**: Proje başlangıcında güvenlik gereksinimlerini açıkça belirlenmesi.

5-) Security Misconfiguration (Güvenlik Yapılandırma Hatası)

Güvenlik Yapılandırma Hatası Nedir?

Güvenlik ayarları uygulanmadığında veya hatalarla dağıtıldığında bir güvenlik yanlış yapılandırması meydana gelebilir. Bu hatalar, uygulamayı ve verilerini bir siber saldırıya veya olası bir ihlale maruz bırakan güvenlik açıkları oluşturur.

Bu hatalar uygulama yığınının herhangi bir düzeyinde meydana gelebilir, örneğin:

- Web veya uygulama sunucuları
- Veritabanları
- Ağ hizmetleri
- Özel kod
- Geliştirme platformları ve çerçeveleri
- Depolama
- Sanal makineler
- Bulut konteynerleri

Örneğin

Bir web sunucusu, varsayılan ayarlarla kurulmuş olabilir. Örneğin, sunucunun hata mesajları ve sistem bilgilerini detaylı olarak gösteren hata sayfaları, saldırganlara sunucu hakkında bilgi verebilir. Bu bilgiler, saldırganların sistemin zayıf noktalarını belirlemesine yardımcı olabilir.

Bir uygulamanın konfigürasyon dosyası, hassas bilgileri (örneğin, veritabanı bağlantı dizeleri, API anahtarları) içeriyor olabilir. Ancak bu dosya, yanlışlıkla herkese açık bir şekilde veya erişim izni olmayan bir yere yüklenmiş olabilir. Bu durumda, saldırganlar bu hassas bilgilere erişebilir ve uygulamanın güvenliğini tehlikeye atabilir.

Güvenlik Yapılandırmalarındaki Hatalar Nelere Yol Açabilir?

Yapılandırma hatalarının çoğu, sistem yöneticilerinin kurulum sırasında bir cihazın veya uygulamanın varsayılan yapılandırmasını (aynı zamanda "hazır" hesap ayarları olarak da bilinir) değiştirmemesi nedeniyle meydana gelir. Bu hata sorunludur çünkü birçok otomatik saldırı, bir hedefin sistemlerinin varsayılan ayarları kullanıp kullanmadığını test ederek başlar. Bu ayarları değiştirerek, kuruluşlar bu tür saldırıların başarılı olma şansını azaltabilir.

Örneğin, bir sistem yöneticisinin bir CMS uygulamasında varsayılan yapılandırmayı koruduğunu düşünün. Varsayılan ayarları değiştirmeden, cihaz, uygulama, ağ ve sistem bu ayarları bilen bir saldırganın istismarlarına karşı savunmasızdır. Bazı tehdit araştırma kuruluşları, yanlış yapılandırmaların bir saldırıya yol açabilecek en önemli 10 istismar arasında olduğunu tahmin ediyor.

Düzeltilmemiş kusurlar

Tüm yazılımların kusurları vardır, ancak çoğu yazılım satıcısı kısa sürede bu kusurları onarmak için yamalar yayınlar. Yamayı yüklemediğinizde, kusurları zaten bilen saldırganlar sistemlerinize sızabilir.

Kullanılmayan sayfalar ve gereksiz servis

Kullanılmayan web sayfaları ve gereksiz özellikler veya hizmetler ayrıca saldırganların bir kurumsal uygulamaya veya cihaza yetkisiz erişim elde etmesine olanak tanır. Bu sorunlar, kontrol edilmediği takdirde komut enjeksiyonları, kaba kuvvet saldırıları ve kimlik bilgisi doldurma istismarları gibi siber saldırılara neden olabilir.

Yetersiz erişim kontrolleri

Tehdit aktörleri, varsayılan parolaları, terk edilmiş kullanıcı hesaplarını veya yöneticilerin güncellemediği veya kaldırmadığı kullanılmayan erişim izinlerini kullanarak ağ altyapısına girebilir. Aşırı izin verici erişim kuralları, saldırganların kötü amaçlı yazılım saldırıları ve veri ihlali dahil olmak üzere kaos yaratmasına da olanak tanır.

Korunmasız dosyalar ve dizinler

Güçlü güvenlik kontrolleriyle korunmayan dosyalar ve dizinler siber saldırılara karşı savunmasızdır. Bilgisayar korsanları, değerli sistem bilgilerini toplamak ve hedefli saldırılar düzenlemek için kolay tahmin edilebilir adlar ve konumlar kullanan platformları ve uygulamaları belirleyebilir.

Öngörülebilir dosya adları ve konumları ayrıca yönetici arayüzlerini açığa çıkarabilir ve saldırganın ayrıcalıklı erişim, yapılandırma ayrıntıları veya iş mantığı elde etmesine ve hatta uygulama işlevlerini eklemesine, kaldırmasına veya değiştirmesine olanak tanıyabilir.

Kötü kodlama uygulamaları ve savunmasız XML dosyalarının kullanımı

Java web.xml dosyalarında birçok güvenlik yanlış yapılandırması meydana gelebilir. Özel hata sayfaları veya SSL yapılandırılmamış olabilir veya kodda web tabanlı erişim kontrolleri eksik olabilir.

Kodlama hataları saldırganların web uygulamalarının bazı bölümlerine SSL olmayan yollarla erişmelerine ve oturum ele geçirme saldırıları başlatmalarına olanak tanıyabilir. Oturum izleme için URL parametreleri kullanmak veya oturum zaman aşımı ayarlamamak da bu saldırılara yol açabilir. Benzer şekilde, HttpOnly bayrağı olmayan çerezler, siteler arası betik çalıştırma (XSS) saldırıları olasılığını artırabilir.

Devre dışı bırakılmış antivirüs

Bazen kullanıcılar, antivirüs belirli bir eylemi, örneğin bir yükleyiciyi çalıştırmayı geçersiz kılarsa antivirüsü geçici olarak kapatır. Kullanıcı kurulumu tamamladıktan sonra, antivirüsü yeniden etkinleştirmeyi unutursa, bu durum kuruluşu saldırılara ve veri ihlallerine karşı savunmasız bırakır.

Yetersiz donanım yönetimi

Bilgisayar korsanları, güvenli olmayan bağlantı noktalarını, aşırı izin verici ağ trafiği kurallarını ve yetersiz yamalanmış ve bakımı yapılmış donanımları kullanarak kurumsal uygulamalara ve verilere erişmek için yönlendiriciler, anahtarlar ve uç noktalar gibi cihazları kullanır.

Nasıl Önlenir?

- Standart Güvenlik Yapılandırmaları Kullanılmalı
- Varsayılan Ayarları Değiştirilmeli
- Güvenlik Duvarları ve Erişim Kontrolü
- Güvenlik Denetimleri ve Testler
- Erişim Kontrolü
- Dokümantasyon ve Değişim Yönetimi
- Otomasyon ve Konfigürasyon Yönetimi Araçları
- Eğitim ve Farkındalık

6-) Vulnerable and Outdated Components (Savunmasız ve Güncel Olmayan Bileşenler)

Nedir?

Güvenlik açığı bulunan ve güncelliğini yitirmiş bileşenler, açık kaynaklı veya tescilli kodun yazılım güvenlik açıkları içermesi veya artık sürdürülmemesi anlamına gelir. Bu kod, kütüphaneler veya çerçeveler biçiminde olabilir ve web uygulamaları için buna Laravel (PHP), Angular (JavaScript), Django (Python) ve diğerleri dahil olabilir. Ne yazık ki, bu kod genellikle güvenlik için çok az veya hiç dikkate alınmadan uygulanır ve bu da uygulama kullanıcıları için potansiyel olarak ağır sonuçlara yol açar ve şirketlerin itibarını riske atar.

Sıfırıncı gün zafiyetleri bazen üçüncü taraf bileşenlerde keşfedilse ve hassas sistemleri ihlal etmek için kullanılsa da , çoğu ihlal BT profesyonelleri tarafından zaten iyi bilinen zayıflıklardan kaynaklanır. Ne

yazık ki, sorunu düzeltmek oldukça karmaşık olabilir ve bir güncelleme komutu çalıştırmak veya güncellenmiş paketleri indirmek kadar basit değildir.

Neden Kaynaklanır?

- **Yetersiz Güncelleme Yönetimi**: Bileşenlerin ve kütüphanelerin güncellemeleri düzenli olarak yapılmazsa, bilinen güvenlik açıkları ve hatalar sistemde kalabilir.
- **Bileşenlerin Yanlış Yönetimi**: Yazılım geliştirme sürecinde kullanılan üçüncü taraf bileşenler ve kütüphanelerin güvenlik durumu yeterince kontrol edilmez.
- **Düşük Farkındalık**: Güvenlik açıklarının farkında olmamak veya bileşenlerin güncellemeleri hakkında bilgi sahibi olmamak.
- **Bağımlılık Yönetimi Eksiklikleri**: Yazılım projelerindeki bağımlılıkların yönetilmemesi veya güncel tutulmaması.

Türleri

- **Eski Yazılım Kütüphaneleri**
- **Güvenlik Açığı İçeren Bileşenler**
- **Desteklenmeyen Bileşenler**

Nasıl Önlenir?

- **Düzenli Güncellemeler ve Yamanın Uygulanması**: Yazılım bileşenlerinin ve kütüphanelerinin düzenli olarak güncellenmesi ve güvenlik yamalarının zamanında uygulanması sağlanmalıdır.
- **Güvenlik Açığı İzleme**: Kullanılan bileşenlerin güvenlik açıkları düzenli olarak izlenmeli ve bilinen güvenlik açıklarına yönelik yamalar derhal uygulanmalıdır.
- **Bağımlılık Yönetimi Araçları Kullanma**: Projelerde kullanılan bağımlılıkların güncel olup olmadığı ve güvenlik açıkları düzenli olarak kontrol edilmelidir. Bu amaçla bağımlılık yönetimi araçları kullanılmalı ve örneğin OWASP Dependency-Check gibi araçlar kullanılarak tarama yapılmalıdır.
- **Güvenli Bileşen Seçimi**: Güvenlik açısından değerlendirilmiş ve desteklenen bileşenlerin seçilmesine özen gösterilmelidir.

7-Identification and Authentication Failures (Tanımlama ve Kimlik Doğrulama Hataları)

Nedir

Tanımlama hataları, adından da anlaşılacağı üzere, sistemin kullanıcıyı tanımlama yeteneğinin eksikliğini ima eder. Benzer şekilde, kimlik doğrulama hataları, uygulamanın kullanıcının kimliğini kendi kimliği olarak doğrulama yetersizliğini ima eder.

Buna eşlik edecek bir örneği ele alalım: Bir kullanıcının oturum açma ekranında yanlış bir kullanıcı adı girmesi ancak web sitesine başarılı bir şekilde girmesi, kimlik tespiti başarısızlığının başlıca örneğidir.

Uygulama, kayıtlı bir kullanıcıyı tanımak için bir kullanıcı adı veya kullanıcı kimliği kullanır.

Buna karşılık, önerilen istemde yanlış parolayı giren ancak web sitesine giriş yapabilen bir kullanıcı, kimlik doğrulama hatasının bir örneğidir. Bu örnekte, uygulama kullanıcının kimliğini doğru bir şekilde doğrulamayı başaramadı ve hatalı kimliğin giriş yapmasına izin verdi.

Örneğin

Bir web uygulamasında kullanıcıların şifrelerini unuttuğunda "şifre sıfırlama" özelliği bulunuyor. Ancak, uygulama sıfırlama isteği yaparken kullanıcıyı doğru bir şekilde kimliklendirmiyor. Örneğin, sistem yalnızca e-posta adresi isteyip şifre sıfırlama bağlantısını bu e-posta adresine gönderiyor. Bu durumda, eğer saldırgan bir kullanıcının e-posta adresini öğrenmişse, şifre sıfırlama bağlantısını elde edebilir ve kullanıcının hesabını ele geçirebilir.

Kaynaklanma Nedenleri

- **Zayıf Şifre Politikaları**: Güçlü şifre gereksinimlerinin olmaması, kullanıcıların kolay tahmin edilebilir veya zayıf şifreler kullanmalarına yol açar.
- **Eksik İki Faktörlü Kimlik Doğrulama (2FA)**: Kimlik doğrulama işlemlerinde yalnızca tek bir faktör (örneğin, şifre) kullanılması, ek bir güvenlik katmanının eksik olması.
- **Yetersiz Oturum Yönetimi**: Oturum süresinin yönetilmemesi, kullanıcıların oturumlarının uzun süre açık kalması veya oturum sonlandırma işlemlerinin yetersiz olması.
- **Kimlik Doğrulama Hataları**: Hatalı veya yetersiz kimlik doğrulama mekanizmaları, örneğin, şifrelerin açık metin olarak saklanması veya yeterince güvenli olmayan kimlik doğrulama yöntemlerinin kullanılması.
- **Yanlış Yapılandırılmış Erişim Kontrolleri**: Erişim kontrol listelerinin veya rol tabanlı erişim kontrollerinin yanlış yapılandırılması, yetkisiz kullanıcıların hassas bilgilere erişmesine neden olabilir.

Türleri

- **Şifre Güvenliği Açıkları**: Zayıf veya tahmin edilebilir şifrelerin kullanılması, şifrelerin güvenli bir şekilde saklanmaması.
- **Kimlik Doğrulama Bypass**: Kimlik doğrulama süreçlerinin atlanması veya aşılması, yetkisiz kullanıcıların sisteme giriş yapabilmesi.
- **Oturum Yönetimi Sorunları**: Oturum çerezlerinin veya tokenlarının yeterince güvenli olmaması, oturum sürelerinin uygun şekilde yönetilmemesi.
- **Eksik İki Faktörlü Kimlik Doğrulama (2FA)**: İki faktörlü kimlik doğrulama mekanizmalarının uygulanmaması, ek bir güvenlik katmanının eksikliği.

Nasıl Önlenir?

- **Güçlü Şifre Politikaları Uygulama**: Kullanıcıların güçlü ve karmaşık şifreler kullanmalarını teşvik etmek amacıyla şifre politikaları oluşturulmalıdır. Şifrelerin belirli bir uzunluk ve karmaşıklık standartlarına uyması sağlanmalıdır.

- **İki Faktörlü Kimlik Doğrulama (2FA)**: İki faktörlü kimlik doğrulama yöntemleri uygulanmalıdır. Kullanıcıların kimliklerini doğrulamak için ek bir doğrulama faktörü (örneğin, SMS kodu veya doğrulama uygulaması) kullanılmalıdır.
- **Güvenli Oturum Yönetimi**: Oturum yönetim süreçleri güvenli bir şekilde yapılandırılmalıdır. Oturum süreleri sınırlandırılmalı, oturum kapama işlemleri etkinleştirilmeli ve oturum çerezleri güvenli bir şekilde yönetilmelidir.
- **Güvenli Kimlik Doğrulama Mekanizmaları**: Kimlik doğrulama mekanizmalarının güvenli ve güncel olduğundan emin olunmalıdır. Şifrelerin hashleme ve şifreleme yöntemleri ile güvenli bir şekilde saklandığı doğrulanmalıdır.
- **Erişim Kontrolleri ve İzinler**: Erişim kontrol politikaları doğru bir şekilde yapılandırılmalı ve kullanıcıların yalnızca yetkili oldukları kaynaklara erişimlerinin sağlanması gerekmektedir.

8-) Software and Data Integrity Failures (Yazılım ve Veri Bütünlüğü Arızaları)

Nedir?

Yazılım ve veri bütünlüğü hataları, bir saldırganın yetkisiz bir şekilde verileri değiştirebilmesi veya silebilmesi durumunda ortaya çıkar. Bu, yazılımdaki güvenlik açıkları veya kötü kodlama uygulamaları nedeniyle olabilir. Saldırganlar, hassas bilgilere erişmek veya sisteme zarar vermek için bu güvenlik açıklarından yararlanabilir.

Yazılım ve veri bütünlüğü arızalarına örnekler

- Bir bilgisayar korsanı bir şirketin veri tabanına erişiyor ve müşteri kayıtlarını değiştiriyor.
- Bir saldırgan, veritabanına kötü amaçlı kod enjekte etmek için bir web uygulamasındaki bir güvenlik açığından yararlanır.
- Kötü niyetli bir içeriden kişi kişisel çıkar sağlamak amacıyla finansal kayıtları değiştirir.

Örneğin

Bir web uygulamasının güncelleme süreci, güncellenmiş yazılım paketlerini indirmek için güvenilir bir kaynaktan doğrulama yapmadan indiriyor olabilir. Saldırgan, yazılım güncellemesi olarak kötü amaçlı bir dosya yükleyebilir ve kullanıcıların sistemlerine zarar verebilir. Bu tür bir durum, yazılım ve veri bütünlüğünün sağlanamadığını gösterir.

Bir veritabanında saklanan veriler, yetkisiz kişiler tarafından değiştirilebilir. Örneğin, veritabanındaki veri bütünlüğü yeterince korunmuyor olabilir ve saldırganlar verileri değiştirebilir, ekleyebilir veya silebilir. Bu durum, verilerin güvenilirliğini ve doğruluğunu tehlikeye atar.

Kaynaklanma Nedenleri

- **Zayıf Veri Doğrulama**: Kullanıcı verilerinin veya sistem girdilerinin yeterince doğrulanmaması, veri bütünlüğü problemlerine yol açabilir.

- **Kötü Kodlama Pratikleri**: Yazılım geliştirme sürecinde güvenlik açıkları veya hatalı kodlama uygulamaları, veri bütünlüğü sorunlarına neden olabilir.
- **Eksik veya Yetersiz Güvenlik Kontrolleri**: Yazılımın veya verilerin güvenliğini korumak için yeterli güvenlik önlemlerinin alınmaması.
- **Hatalı Güncellemeler ve Yamanın Uygulanması**: Yazılım güncellemeleri ve yamaların hatalı bir şekilde uygulanması, yazılımın veya verinin bütünlüğünü bozabilir.
- **Yetkisiz Erişim**: Yetkisiz kullanıcıların yazılım veya verilere erişim sağlayarak bunları değiştirmesi veya manipüle etmesi.

Yazılım ve veri bütünlüğü arızalarının etkisi

Yazılım ve veri bütünlüğü arızalarının etkisi ciddi olabilir. Finansal kayıplara, itibar kaybına, yasal yükümlülüklere, müşteri güveninin kaybına vb. neden olabilir.

Yazılım ve veri ihlallerine ilişkin istatistikler

IBM'in Veri İhlali Maliyeti Raporu 2020'ye göre, küresel olarak bir veri ihlalinin ortalama maliyeti 3,86 milyon dolardı. Rapor ayrıca kuruluşların bir ihlali tespit edip sınırlamasının ortalama 280 gün sürdüğünü buldu.

Türleri

- **Veri Bozulması**: Verilerin yanlışlıkla veya kötü niyetli bir şekilde değiştirilmesi, silinmesi veya bozulması.
- **Yazılım Bozulması**: Yazılım bileşenlerinin veya kodunun hatalı bir şekilde değiştirilmesi veya bozulması.
- **Eksik Veri Doğrulama**: Kullanıcı girişleri veya diğer verilerin yeterince doğrulanmaması, veri bütünlüğü sorunlarına yol açabilir.
- **Yazılım Güncelleme Sorunları**: Yazılım güncellemelerinin hatalı uygulanması veya uyumsuzluklar nedeniyle oluşan veri bütünlüğü problemleri.

Nasıl Önlenir?

- **Veri Doğrulama ve Temizleme**: Kullanıcı verilerinin ve sistem girdilerinin doğru ve güvenli bir şekilde doğrulanmasını ve temizlenmesini sağlayacak süreçler uygulanmalıdır. Verilerin beklenmedik değişikliklere karşı korunması sağlanmalıdır.
- **Güvenli Kodlama Pratikleri**: Güvenli yazılım geliştirme pratikleri benimsenmelidir. Kodun güvenliği ve veri bütünlüğü sağlamak amacıyla düzenli kod incelemeleri ve güvenlik testleri yapılmalıdır.
- **Güvenlik Kontrolleri ve İzleme**: Yazılım ve veriler için etkili güvenlik kontrolleri uygulanmalı ve düzenli izleme yapılmalıdır. Yetkisiz erişimlerin ve veri değişikliklerinin tespit edilmesini sağlayacak mekanizmalar oluşturulmalıdır.
- **Doğru Güncellemeler ve Yama Yönetimi**: Yazılım güncellemeleri ve yamalar doğru bir şekilde uygulanmalıdır. Güncellemelerin ve yamaların uyumlu ve güvenli olduğundan emin olunmalıdır.

- **Yetki Yönetimi**: Kullanıcıların ve sistemlerin yalnızca yetkili oldukları kaynaklara erişimlerini sağlamak için etkili yetki yönetimi politikaları uygulanmalıdır. Yetkisiz erişimlerin önlenmesi için uygun kontroller oluşturulmalıdır.
- **Veri Yedekleme ve Kurtarma**: Veri yedekleme ve kurtarma süreçleri düzenli olarak gerçekleştirilmelidir. Veri kaybı veya bozulması durumunda verilerin geri getirilmesini sağlayacak bir kurtarma planı oluşturulmalıdır.

9-) Security Logging and Monitoring Failures (Güvenlik Günlüğü ve İzleme Arızaları)

Güvenlik günlüğü ve izleme hataları, bir sistem veya uygulama güvenlik olaylarını düzgün bir şekilde günlüğe kaydetmeyi veya izlemeyi başaramadığında ortaya çıkabilen güvenlik açıklarıdır. Bu, saldırganların tespit edilmeden sistemlere ve verilere yetkisiz erişim elde etmesine olanak tanıyabilir. Bir sistemin veya uygulamanın güvenlik olaylarını yeterince düzgün bir şekilde kaydetmemesi veya izlememesi durumunu ifade eder. Bu tür arızalar, güvenlik olaylarının tespit edilmesini ve yanıt verilmesini zorlaştırabilir.

Neden Kaynaklanır?

- **Eksik Günlükleme**: Kritik güvenlik olaylarının veya sistem aktivitelerinin yeterince detaylı olarak kaydedilmemesi.
- **Yetersiz İzleme**: Güvenlik olaylarının ve sistem performansının yeterince izlenmemesi, anormal davranışların veya saldırıların tespit edilememesi.
- **Günlüklerin Güvenliğinin Sağlanmaması**: Günlüklerin değiştirilmesine veya silinmesine karşı yeterli güvenlik önlemlerinin alınmaması.
- **Yanlış Yapılandırılmış Günlükleme Sistemleri**: Günlükleme sistemlerinin yanlış veya eksik yapılandırılması, önemli bilgilerin kaybolmasına veya yanlış yerlerde depolanmasına neden olabilir.
- **Sistem Kaynakları Sınırlamaları**: Günlükleme ve izleme sistemleri için yeterli kaynakların ayrılmaması, performans sorunlarına yol açabilir.

**Örneğin

**

Bir şirket, sunucularında güvenlik olaylarını izlemek için bir günlükleme sistemi kurmuş olsun. Ancak, günlükleme sistemi bazı önemli olayları kaydetmiyor veya verileri eksik topluyor. Örneğin, bir siber saldırganın sunucuya izinsiz girdiğini fark etmeyebilirler çünkü ilgili günlük kayıtları yok. Bu tür bir eksiklik, olayın fark edilmeden ilerlemesine neden olabilir.

Bir ağ izleme sistemi, ağ trafiğini analiz ederek şüpheli aktiviteleri tespit etmeye çalışıyor. Ancak, sistem belirli bir tür trafiği göz ardı ediyor veya yanlış yorumluyor. Bu durum, saldırganların bu trafik türünü kullanarak sisteme zarar vermesine yol açabilir.

Türleri

- **Eksik veya Yetersiz Günlükleme**: Sistem etkinliklerinin veya güvenlik olaylarının eksik veya yetersiz bir şekilde kaydedilmesi.
- **Güvenlik Açıkları**: Günlüklerin değiştirilmesi veya silinmesi gibi güvenlik açıklarına karşı korunmaması.
- **Yanlış Konfigürasyon**: Günlükleme ve izleme sistemlerinin yanlış veya eksik yapılandırılması.
- **Yetersiz İzleme**: Güvenlik olaylarının gerçek zamanlı olarak izlenmemesi veya anormal aktivitelerin tespit edilmemesi.

Nasıl Önlenir?

- **Kapsamlı Günlükleme Uygulama**: Kritik güvenlik olayları ve sistem aktiviteleri için kapsamlı günlükleme politikaları oluşturulmalıdır. Günlüklerin detaylı ve doğru bir şekilde kaydedilmesi sağlanmalıdır.
- **Etkili İzleme Sistemleri Kullanma**: Güvenlik olaylarının ve sistem performansının etkili bir şekilde izlenmesini sağlayacak araçlar ve sistemler kullanılmalıdır. Anormal davranışları ve potansiyel saldırıları tespit etmek için izleme sistemleri kurulmalıdır.
- **Günlük Güvenliğini Sağlama**: Günlüklerin güvenli bir şekilde saklanması için uygun güvenlik önlemleri alınmalıdır. Günlüklerin değiştirilmesi veya silinmesine karşı koruma sağlanmalıdır.
- **Doğru Yapılandırma**: Günlükleme ve izleme sistemlerinin doğru ve eksiksiz bir şekilde yapılandırıldığından emin olunmalıdır. Konfigürasyon hataları önlenmelidir.
- **Kaynak Yönetimi**: Günlükleme ve izleme sistemleri için yeterli kaynak ayrılmalı ve performans sorunlarına karşı önlemler alınmalıdır.
- **Düzenli İnceleme ve Analiz**: Günlükler düzenli olarak incelenmeli ve analiz edilmelidir. Güvenlik olayları hakkında raporlar hazırlanmalı ve olası tehditler belirlenmelidir.

10-) Server-Side Request Forgery (Sunucu Taraflı İstek Sahteciliği)

Server-Side Request Forgery (SSRF - Sunucu Taraflı İstek Sahteciliği), bir saldırganın sunucunun, istemci yerine kendi tarafında yapacağı istekleri kötüye kullanmasını ifade eden bir güvenlik açığıdır. Bu tür bir saldırıda, kötü niyetli bir kullanıcı, sunucunun arka uç sistemlere veya yerel ağ kaynaklarına yetkisiz erişim sağlamasını sağlayabilir.

Neden Kaynaklanır?

1. **Yetersiz Girdi Doğrulama**: Kullanıcı tarafından sağlanan verilerin yeterince doğrulanmaması veya filtrelenmemesi, saldırganların kötü amaçlı istekler göndermesine neden olabilir.
2. **İçsel İsteklerin Yönetiminde Hatalar**: Sunucunun iç kaynaklarına veya diğer sistemlerine yapılacak isteklerin kontrolsüz bir şekilde yönetilmesi.

3. **Güvenlik Duvarı ve Erişim Kontrollerinin Eksikliği**: Sunucunun iç ağ kaynaklarına erişimi kısıtlayan uygun güvenlik duvarı veya erişim kontrollerinin olmaması.
4. **Yetersiz İzinler ve Yetkiler**: Sunucunun yeterli izin ve yetkilere sahip olmaması, bu nedenle iç kaynaklara veya diğer sistemlere erişim sağlaması.

Örneğin

Bir web uygulaması, kullanıcıların URL girerek bazı bilgileri almasını sağlayan bir özelliğe sahip olsun. Bu özellik, kullanıcının verdiği URL'ye HTTP istekleri göndererek veri çeker. Ancak uygulama, kullanıcının verdiği URL'yi doğrudan kullanmadan önce yeterince denetim yapmıyor.

Bir saldırgan, bu URL özelliğini kullanarak sunucunun yerel ağındaki hassas bilgileri sızdırabilir. Örneğin, saldırgan şöyle bir URL girebilir: `http://localhost/admin`. Sunucu, bu isteği yerel ağdaki `/admin` sayfasına yönlendirir ve eğer bu sayfa hassas bilgiler içeriyorsa, saldırgan bu bilgilere erişebilir.

Türleri

1. **İç Ağa Erişim**: Saldırganın sunucu aracılığıyla iç ağdaki sistemlere veya hizmetlere erişim sağlaması.
2. **Gizli Bilgi Sızıntısı**: Sunucunun iç kaynaklardan veya hizmetlerden gizli bilgi alması ve bu bilgileri saldırgana sunması.
3. **Kötü Amaçlı İç İstekler**: Sunucu üzerinden kötü amaçlı istekler yaparak diğer sistemlerde zararlı işlemler gerçekleştirilmesi.
4. **Yoklama ve Tarama**: Saldırganın sunucuyu kullanarak iç ağdaki diğer sistemleri taraması veya yoklaması.

Nasıl Önlenir?

- **Girdi Doğrulama ve Filtreleme**
- **İç Kaynaklara Erişim Kontrolleri**
- **İç İsteklerin Sınırlandırılması**
- **Erişim Yetkileri Yönetimi**
- **Güvenlik Testleri ve Tarama**
- **İzleme ve Denetleme**

Hazırlayan: Ayberk İlbaş

Linkedin

Github