OWASP TOP TEN

A01:2021-Broken Access Control

Nedir?

Erişim kontrolü (veya yetkilendirme), teşebbüs edilen eylemleri veya talep ettikleri kaynaklara erişimi kimin (veya neyin) gerçekleştirebileceğine ilişkin kısıtlamaların uygulanmasıdır. Web uygulamaları bağlamında erişim kontrolü, kimlik doğrulama ve oturum yönetimine bağlıdır:

Kimlik doğrulama , kullanıcıyı tanımlar ve söyledikleri kişi olduklarını onaylar.
Oturum yönetimi , aynı kullanıcı tarafından hangi sonraki HTTP isteklerinin yapıldığını tanımlar.
Erişim denetimi , kullanıcının gerçekleştirmeye çalıştığı eylemi gerçekleştirmesine izin verilip verilmediğini belirler.
Bozuk erişim denetimleri, yaygın olarak karşılaşılan ve genellikle kritik bir güvenlik açığıdır. Erişim kontrollerinin tasarımı ve yönetimi, teknik bir uygulamaya iş, organizasyon ve yasal kısıtlamalar uygulayan karmaşık ve dinamik bir sorundur. Erişim kontrolü tasarım kararları teknoloji tarafından değil insanlar tarafından verilmelidir ve hata potansiyeli yüksektir.

Kullanıcı açısından bakıldığında, erişim kontrolleri aşağıdaki kategorilere ayrılabilir:

Dikey erişim kontrolleri,Yatay erişim kontrolleri,Bağlama bağlı erişim kontrolleri

Dikey erişim kontrolleri, diğer kullanıcı türleri için mevcut olmayan hassas işlevlere erişimi kısıtlayan mekanizmalardır.

Dikey erişim kontrolleri ile farklı tipteki kullanıcılar, farklı uygulama işlevlerine erişebilir. Örneğin, sıradan bir kullanıcının bu işlemlere erişimi yokken bir yönetici herhangi bir kullanıcının hesabını değiştirebilir veya silebilir. Dikey erişim kontrolleri, görevler ayrılığı ve en az yetki gibi iş politikalarını uygulamak için tasarlanmış güvenlik modellerinin daha ayrıntılı uygulamaları olabilir.

Yatay erişim kontrolleri

Yatay erişim kontrolleri, kaynaklara erişimi, bu kaynaklara erişmelerine özel olarak izin verilen kullanıcılarla kısıtlayan mekanizmalardır.

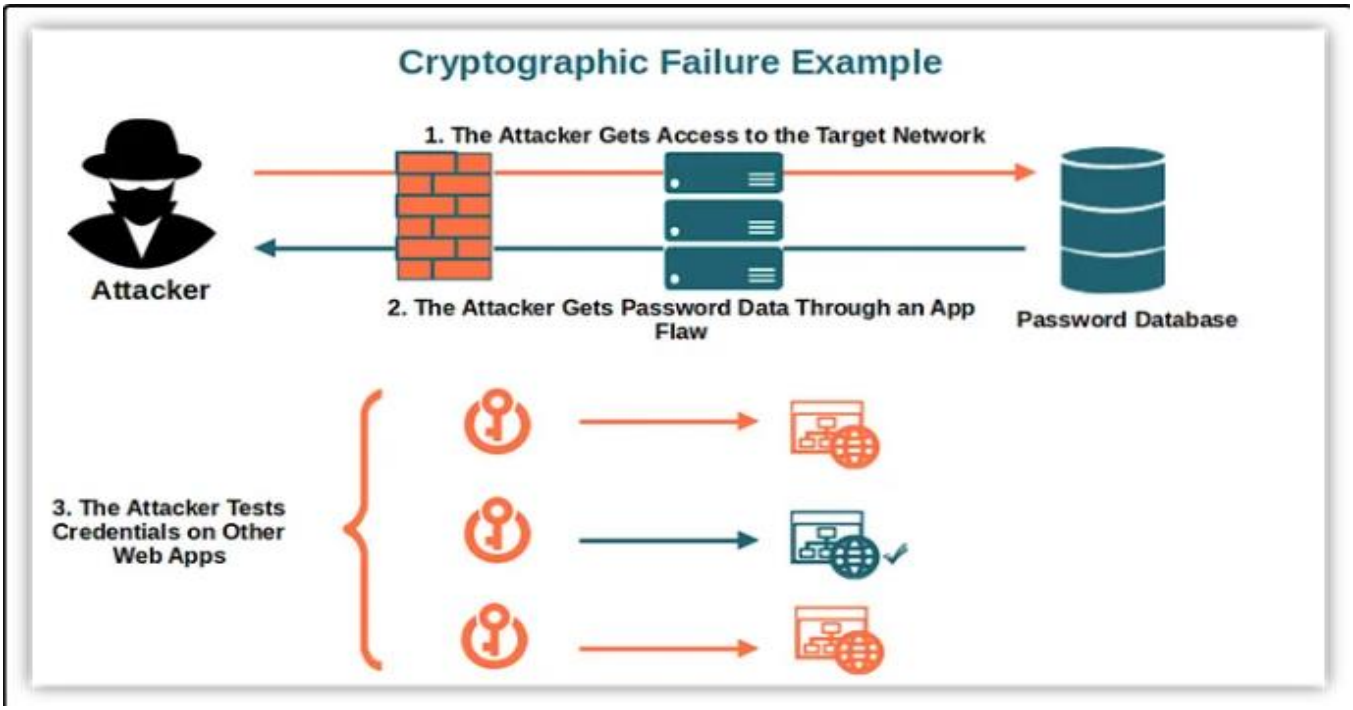
Yatay erişim denetimleriyle, farklı kullanıcılar aynı türdeki kaynakların bir alt kümesine erişebilir. Örneğin, bir bankacılık uygulaması, bir kullanıcının işlemleri görüntülemesine ve kendi hesaplarından ödeme yapmasına izin verir, ancak başka bir kullanıcının hesaplarına izin vermez.

Bağlama bağlı erişim kontrolleri

Bağlama bağlı erişim kontrolleri, uygulamanın durumuna veya kullanıcının onunla etkileşimine bağlı olarak işlevlere ve kaynaklara erişimi kısıtlar.

Bağlama bağlı erişim kontrolleri, bir kullanıcının eylemleri yanlış sırada gerçekleştirmesini engeller. Örneğin, bir perakende web sitesi, kullanıcıların ödeme yaptıktan sonra alışveriş sepetlerinin içeriğini değiştirmesini engelleyebilir.

A02:2021-Cryptographic Failures



Nedir?

Bu zafiyet türü verilerin şifrelenmemesi veya şifrelenen verilerin eski, varsayılan, açığa çıkmış şifreleme algoritmaları kullanılmasından ortaya çıkıyor. Geçerliliğini yitirmiş algoritmalar veya şifrelenmeden iletilen datalar zafiyet oluşmasına sebep olmaktadır.

Kredi kartı bilgileri, parolalar, kişisel kayıtlar, firma belgeleri ve daha birçok önemli bilgilerin şifrelenmesi gerekmektedir.

Saldırgan, sql injection zafiyetini kullanarak veritabanındaki verilere erişebilse bile şifrelenmiş verileri anlamlandıramaz ise bir işine yaramayacaktır. Kullanılmakta olan şifreleme algoritmalarının güçlü olması burada işimize yarayacaktır.

Ayrıca sunucu tarafında bilgiler kayıt altına alınırken şifrelense bile, bilgiler kullanıcıdan sunucuya aktarılırken şifrelenmeden gidiyor ise buda bir zafiyet oluşturmaktadır. Saldırgan kullanıcı ile sunucu arasındaki trafiği izleyip önemli bilgilere erişebilir. Bu yüzden sunucu tarafında şifrelemenin yanı sıra transfer esnasında da şifreli data akışı (https, ssl vpn vb.) kullanılmalıdır.

Bunları önlemek için:

Saklanan veriler sınıflandırılmalı

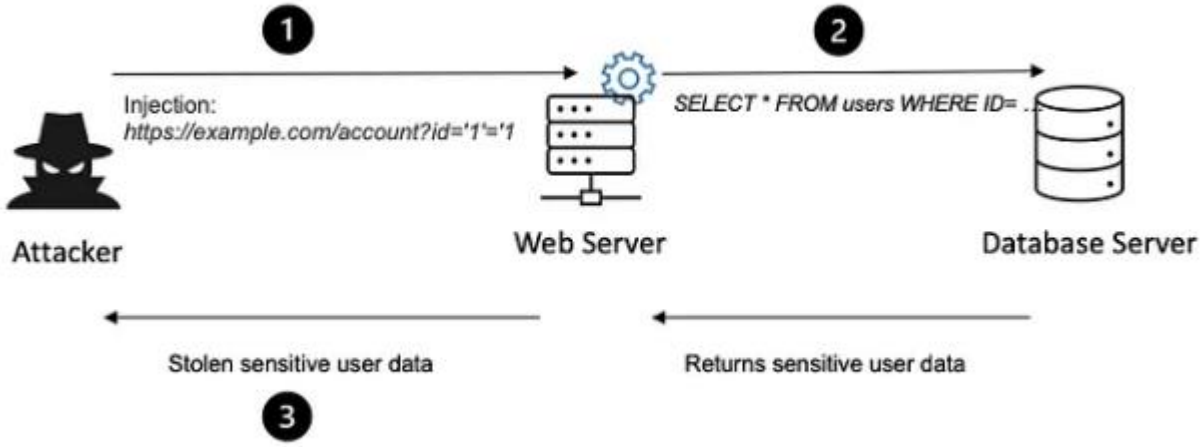
Hassas olarak sınıflandırılmış gereksiz veriler saklanmamalı

Gerekli hassas veriler şifrelenerek saklanmalı

Güçlü şifreleme algoritmaları kullanılmalı

Anahtarların güvenliği sağlanmalı

A03:2021-Injection



Nedir?

Injection zafiyetleri genellikle kullanıcıdan alınan, kontrol edilmeyen ya da önlem alınmayan verilerin komut olarak çalıştırılması ya da sorguya dahil edilmesi yüzünden oluşan zafiyetlerdir. İstatistiklere göre, şirketlerin % 28'i bu zafiyete maruz kalmaktadır.

Bu güvenlik açığı aşağıdaki saldırı vektörlerine bölünmüştür:

- **SQL**, **LDAP**, **XPath** sorguları aracılığıyla enjeksiyon
- İşletim Sistemlerindeki komutlarla enjeksiyon
- **XML** ayrıştırma yoluyla enjeksiyon
- **SMTP** başlıkları aracılığıyla enjeksiyon

Saldırgan bu vektörleri (*farklı bir veri göndererek sistemde komut çalıştırabilir*) kullanarak hem bir hesaba hem de erişmemesi gereken bu kaynağın tüm istemci veri tabanına erişim sağlayabilir. SQL veri tabanı türüne bağlı olarak farklı söz dizimi kullanarak veri tabanıyla çalışmak için yalnızca özel karakterler ve ek operatörler kullanarak hak yükseltme zafiyetini kullanmış olur. Sistemde ki etkinliğini artırır, hatta bütün organizasyona sızabilir. Bu zafiyet kullanıcıdan gelen verinin filtrelenmemesi veya kötü kodlardan temizlenmemesi sonucu ortaya çıkar.

Injection açığından korunmak için alınması gereken önlemler:

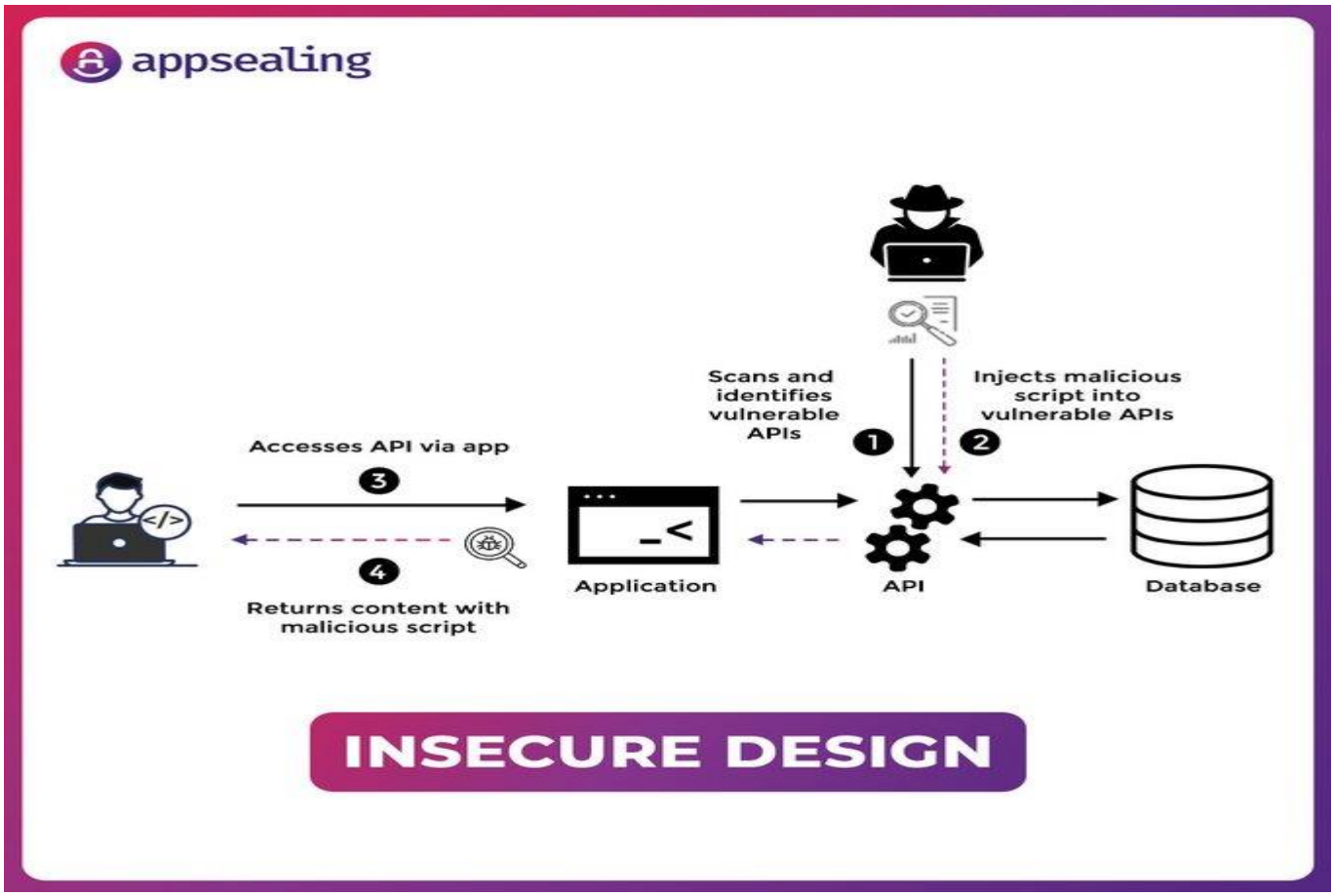
· Parametrelerin Doğrulanması

· SQL Parametreleştirme

· Kodlama Standartları

· WAF Kullanımı

A04:2021-İnsecure Design



Nedir?

Insecure Design (Güvensiz Tasarım), bir web uygulamasının tasarımında yapılan hatalar veya eksiklikler nedeniyle ortaya çıkan bir güvenlik açığıdır. Bu, uygulamanın tasarımında yapılan hataların, uygulamanın tüm yaşam döngüsü boyunca devam etmesine ve güvenliğini olumsuz etkilemesine neden olabilir. Güvensiz tasarım, uygulamanın özellikle kimlik doğrulama, yetkilendirme, veri gizliliği ve bütünlüğü gibi önemli güvenlik konularında hatalar içermesiyle ortaya çıkabilir.

Insecure Design açığından korunmak için alınması gereken önlemler:

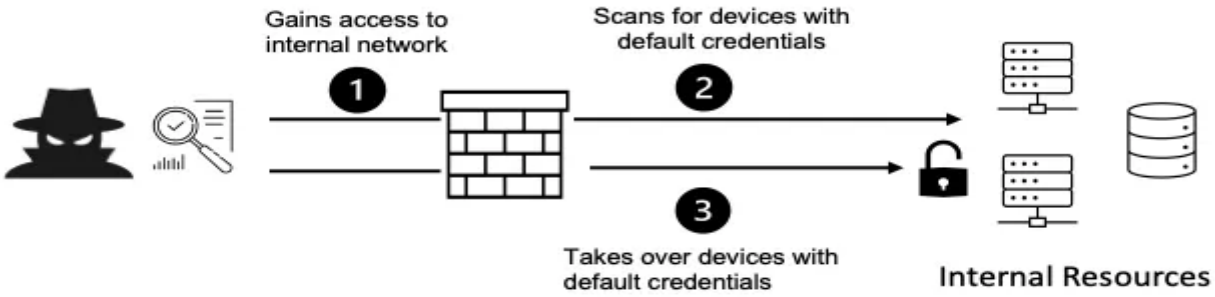
Güvenli tasarım ilkeleri uygulanmalıdır.

Uygulama geliştirme ekibi, güvenlik odaklı bir yaklaşım benimsemelidir.

Güvenlik açıkları düzeltilmelidir.

Uygulama güvenliği için en iyi uygulamalar kullanılmalıdır.

A05:2021-Security Misconfiguration



Nedir?

Bir Web uygulamasının güvenliğini sağlamak, tüm altyapı bileşenlerinin güvenli bir şekilde yapılandırılmasını gerektirir. Uygulama bileşenleri (frameworks gibi), web sunucusu, veritabanı sunucusu ve platformun kendisidir. Sunucu bileşenlerinin varsayılan ayarları genellikle güvensizdir ve saldırılar için fırsatlar yaratır. Örneğin, bir XSS saldırısında JavaScript aracılığıyla bir oturum çerezinin çalınması, varsayılan olarak devre dışı bırakılan cookie_http only ayarı sayesinde mümkün hale gelir.

Sunucu doğru şekilde yapılandırılmış ise ve cookie_httponly seçeneği etkinleştirilmişse, JavaScript aracılığıyla bir oturum tanımlama bilgisi almak imkansızdır, ancak bu basit ve önemli ayar, ödeme sistemlerinin kişisel hesapları gibi kritik yerlerde genellikle eksiktir.

Yanlış Güvenlik Yapılandırması Güvenlik açığına bir başka örnek, Redis (açık kaynak veri yapısı sunucusu), Memcached (genel amaçlı dağıtılmış bir bellek ön bellekleme sistemi) ve diğerleri gibi veritabanı sunucularında varsayılan ayarların kullanılmasıdır. Özel bir hizmete sunucunun genel IP adresinden erişilebilir ve / veya üretici tarafından varsayılan olarak belirlenen parolalar kullanılmıştır. Bu, bir saldırganın, aralarında genellikle oturum çerezlerinin (zaten bildiğimiz) ve tarayıcıda kullanıcılara görüntülenen verilerin (ayrıca bir XSS saldırısının kullanılmasını mümkün kılan) bulunduğu verileri kolayca okuyup değiştirmesine olanak tanır.

Yazılımların her zaman güncel olması gerekir. Güvenlik açıkları her gün çok çeşitli işletim sistemi, web sunucuları, veritabanı sunucuları, posta sunucuları vb. gibi yazılım bileşenlerinde bulunur. Ve uygulamanız düzgün bir şekilde yazılsa ve gelen tüm verileri dikkatlice kontrol etse ve genel olarak iyi korunsa bile, bu, bir gün işletim sisteminizde veya Web sunucunuzda bir güvenlik açığı olmayacağı anlamına gelmez.

Yani uygulamanın herhangi bir seviyesinde yanlış güvenlik yapılandırması meydana gelebilir:

– ağ hizmetleri

– Web sunucusu

– veri tabanı

– önceden yüklenmiş sanal makineler

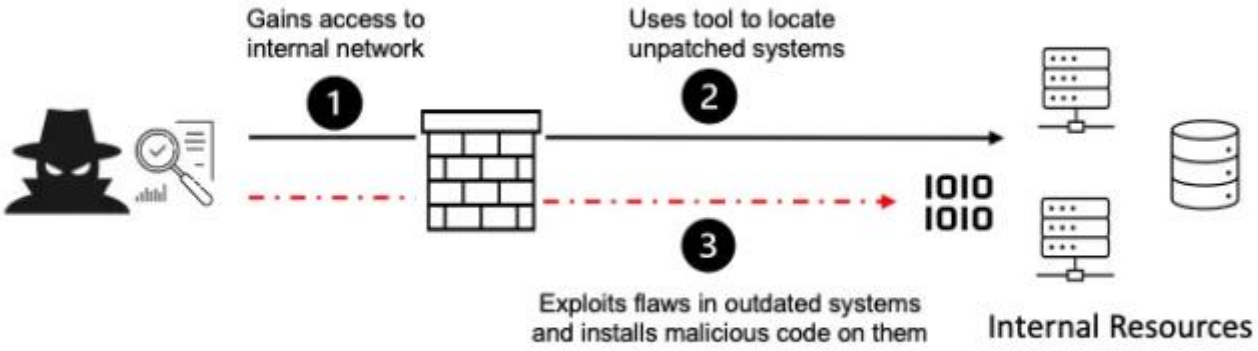
– depolama

Önlem

Güvenliği kontrol etmenin en iyi yolu, yalnızca ağ güvenliğindeki (çoğu tarayıcının yaptığı gibi) yanlış yapılandırmaları algılayan değil, aynı zamanda web uygulamalarının güvenliğine odaklanan profesyonel bir tarayıcı kullanmaktır .

Yanlış güvenlik ayarlarının çok yaygın bir başka nedeni, varsayılan ayarlara güvendir. Profesyonel yazılımın varsayılan olarak korunduğunu düşünmemelisiniz. Web sunucusu, uygulama sunucusu ve veritabanı sunucusu dahil olmak üzere yüklediğiniz her yazılım, manuel güvenlik yapılandırması gerektirir.

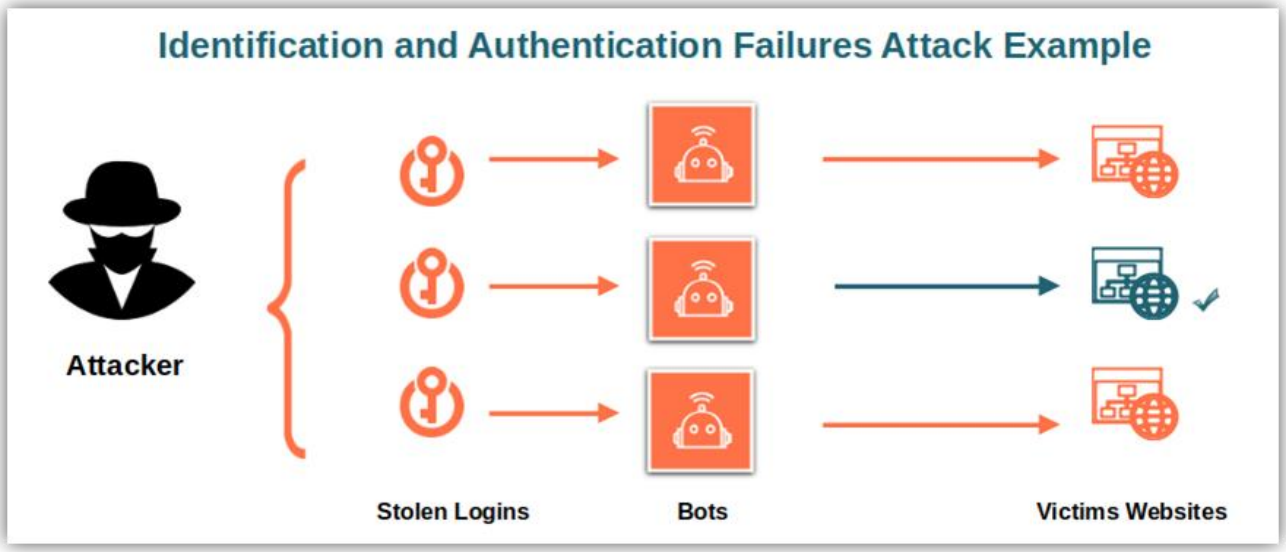
A06:2021-Vulnerable and Outdated Components



Nedir?

Eğer uygulamanız içerisindeki **bileşenlerin versiyonların**ı bilmiyorsanız, o bileşen savunmasız, desteksiz (unsupported) veya tarihi geçmiş (out dated) olabilir. Bu durumda sizin uygulamanızın o **bileşene bağımlılığı** olduğu için uygulamanız da savunmasız kalmış demektir. Ayrıca, uygulamalarımız içerisinde açık kaynak kodlu bileşenler kullanabiliriz ve bu bileşenlerin bazı versiyonlarında güvenlik açıkları olabilir. Bu güvenlik açıkları, uygulamamızı otomatik olarak savunmasız bırakacaktır.Bu sorunlardan kaçınabilmek için; öncelikle bütün varlıklarımızı(assets) bilmemiz gerekmektedir. İkincisi, her bir bileşenin savunmasız olup olmadığını bilmeliyiz ve **proaktif bir şekilde test yapmalıyız**. Üçüncüsü, tarihi geçmiş bileşenlerimizi güncellemeliyiz ve varsa güvenlik için patch atmalıyız.

A07:2021-Identification and Authentication Failures Attack Example



Nedir?

Identification and Authentication Failures, bir kullanıcının kimliğinin doğrulanması veya yetkilendirilmesi sırasında yaşanan sorunlar nedeniyle oluşan bir güvenlik açığıdır. Bu tür bir açık, bir saldırganın bir kullanıcının kimliğini çalmasına veya sahte bir kimlik kullanarak uygulamaya erişmesine izin verebilir.

Örneğin, bir uygulama, kullanıcı adı ve parola kombinasyonu gibi temel kimlik doğrulama yöntemlerini kullanarak kimlik doğrulama yapabilir. Ancak, uygulama bu bilgileri doğru bir şekilde doğrulamazsa, saldırganların sahte kimlik bilgileri kullanarak sisteme giriş yapması mümkündür.

Identification and Authentication Failures açığından korunmak için alınması gereken önlemler:

· Güçlü kimlik doğrulama yöntemleri kullanmak.

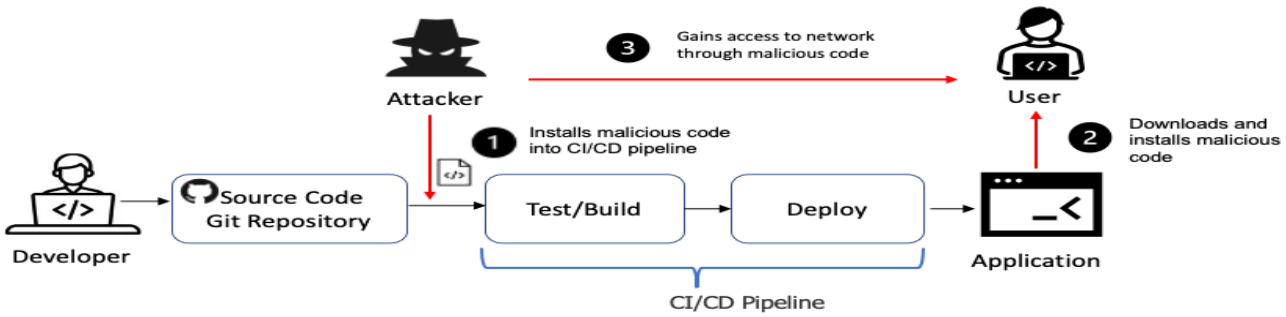
· Kimlik doğrulama işlemlerini izlemek.

· Kimlik bilgilerini şifrelemek.

· Düzenli olarak kimlik doğrulama politikalarını kontrol etmek.

· Çok faktörlü kimlik doğrulama yöntemlerini kullanmak.

A08:2021-Software and Data Integrity Failures



Nedir?

Software and Data Integrity Failures, yazılım veya verinin yetkisiz veya beklenmedik şekilde değiştirilmesi durumunda ortaya çıkan güvenlik açıklarını ifade eder. Bu tür güvenlik açıkları, kötü niyetli kişilerin sistem üzerinde kontrol sahibi olmasına veya hassas bilgileri ele geçirmesine olanak tanır.

Örnekler ve Gerçek Hayattan Senaryolar

SolarWinds Orion Saldırısı: SolarWinds, büyük bir IT yönetim yazılımı sağlayıcısıdır. Aralık 2020'de, SolarWinds'in Orion platformuna yapılan bir tedarik zinciri saldırısı keşfedildi. Saldırganlar, SolarWinds'in güncelleme mekanizmasını manipüle ederek zararlı bir güncelleme sundular. Bu güncelleme, birçok önemli kuruma yayıldı ve ciddi güvenlik ihlallerine neden oldu.
Sonuç: Binlerce kurum etkilendi, saldırganlar uzun süre fark edilmeden sistemlere erişim sağladı.
Önleme: Yazılım güncellemelerinin bütünlük kontrolleri ile doğrulanması ve dijital imzaların kullanılması.

Equifax Veri İhlali:2017'de gerçekleşen bu olayda, Equifax'ın web uygulamasındaki bir güvenlik açığı nedeniyle milyonlarca kişiye ait hassas bilgi çalındı.
Sonuç: Yaklaşık 147 milyon Amerikalının sosyal güvenlik numaraları, doğum tarihleri ve adresleri çalındı.
Önleme: Yazılım güncellemelerinin ve yamaların düzenli olarak uygulanması, güvenlik açığı tarama araçlarının kullanılması.

Önleme Yöntemleri ve Güvenlik Tedbirleri

Dijital İmzalar: Yazılım veya verinin beklenen kaynaktan geldiğini ve değiştirilmediğini doğrulamak için dijital imzalar kullanılır.
Örnek: Bir yazılım güncellemesi indirildiğinde, bu güncellemenin güvenilir bir kaynaktan geldiğini doğrulamak için dijital imza kontrolü yapılmalıdır.

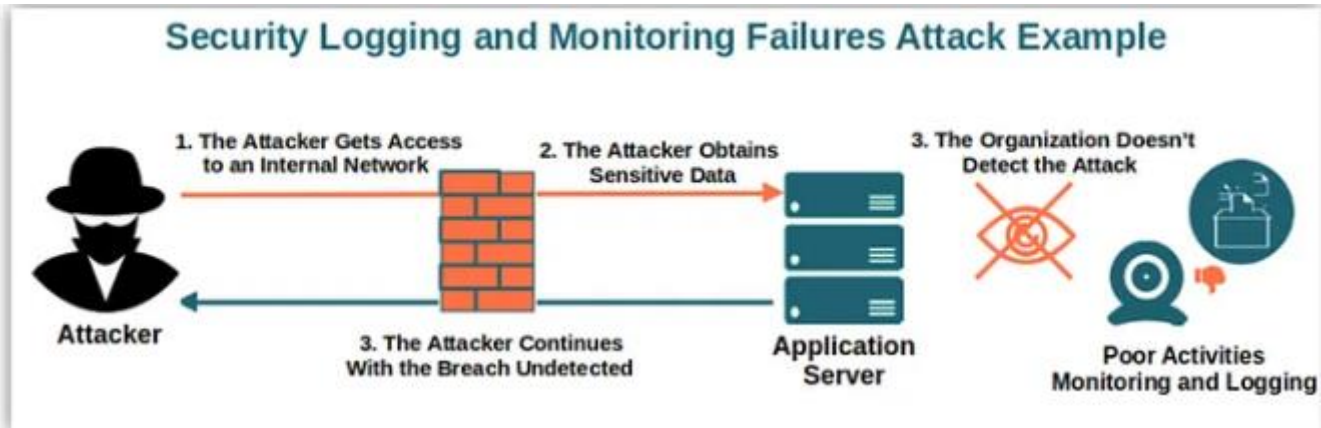
Güvenilir Depolar:Kütüphaneler ve bağımlılıkların güvenilir depolardan alınması gerekir. Daha yüksek risk profiline sahip durumlar için, güvenilir bir iç depo kullanılabilir.
Örnek: Yazılım geliştirme sürecinde kullanılan tüm kütüphanelerin güvenilir kaynaklardan indirildiğini doğrulamak için araçlar (örneğin, OWASP Dependency-Check) kullanmak.

Kod ve Konfigürasyon Değişikliklerinin İncelenmesi: Yazılım hattına kötü niyetli kod veya yapılandırma eklenme riskini en aza indirmek için bir inceleme süreci oluşturulmalıdır.
Örnek: Kod incelemeleri ve otomatik testlerin yapılması, kod değişikliklerinin doğrulanmasını sağlar.

CI/CD Pipeline Güvenliği:Sürekli entegrasyon ve sürekli dağıtıma (CI/CD) uygun şekilde ayrıldığından, yapılandırıldığından ve erişim kontrolüne sahip olduğundan emin olunmalıdır.
Örnek: CI/CD süreçlerinin sadece yetkili kişiler tarafından erişilebilir olmasını sağlamak ve tüm değişikliklerin izlenebilir olmasını sağlamak.

Serileştirilmiş Verilerin Güvenliği:İmzalanmamış veya şifrelenmemiş serileştirilmiş verilerin, bir tür bütünlük kontrolü veya dijital imza olmadan güvenilmeyen istemcilere gönderilmemesi.
Örnek: Web uygulamalarında JSON Web Tokens (JWT) kullanırken, token'ların güvenli bir şekilde imzalandığından emin olmak.

A09:2021-Security Logging and Monitoring Failures



Nedir?

Saldırganlar saldırıya geçmeden önce uzun süreler boyunca vakitlerini sistem hakkında bilgiler elde etmeye ayırırlar. Bilgi elde etme sürecinde pasif veya aktif yöntemler kullanılabilir. Aktif yöntemlerin pasif yöntemlerden farkının karşı tarafla etkileşime geçilmesi olduğunu biliyorduk. Etkileşime geçmek aslında artık karşı taraf tarafından fark edilebilir olmak demektir. Hedef sistemde loglama ve monitoring işlemleri düzgünce yapıldığında saldırganların bilgi toplama süreçleri veya saldırılar erken evrede fark edilip önlenebilirler. Bunun için gerekli her şeyin loglandığından ve monitoring işlemlerinin yapıldığından emin olunmalıdır.

Loglama ve izleme süreçlerindeki eksiklikler OWASP TOP listesinde 9 numarada yer alan Security Logging and Monitoring Failures veya Güvenlik İzleme ve Loglama Hataları'na sebep olur. Loglanması gereken verilerin düzenli olarak loglanmaması şüpheli davranışların tespitini zorlaştırır. Bilgi toplama aşamalarında olduğu gibi brute-force veya dos saldırıları gibi zaman alan ve sistemde çok sayıda log bırakan saldırılar bu adımların atlanması halinde kolaylıkla başarıya ulaşabilir.

Zafiyetten korunmak için:

Loglama ve izleme süreçleri düzgünce yürütülmelidir.

Saldırı olabilecek logları tespit edip bildirebilen IDS/IPS sistemlerine başvurulmalıdır.

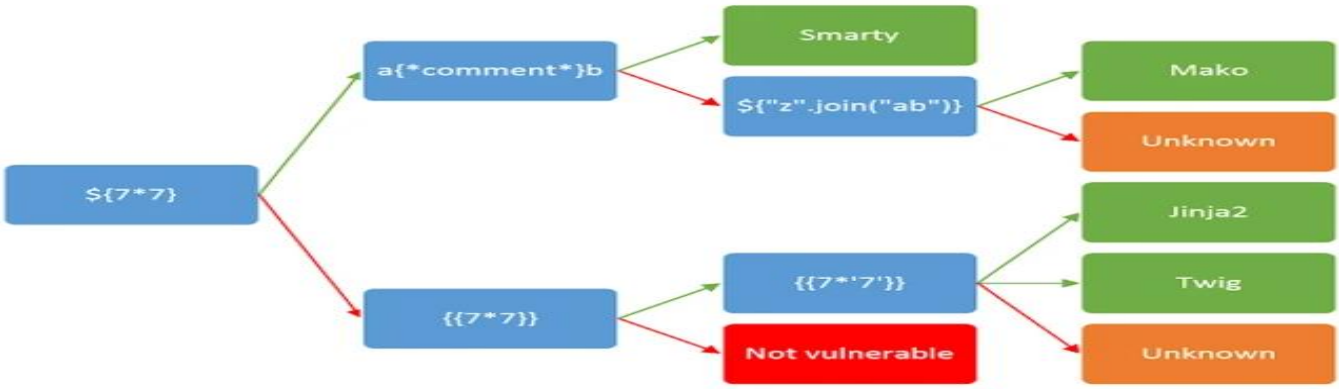
Düzenli sızma testi yaptırılmalıdır.

Loglar yedeklenmelidir.

Gerçek zamanlı alarm oluşturabilen monitoring sistemleri kullanılmalıdır.

Logların bütünlüğü güvence altına alınmalıdır.

A10:2021-Server-Side Request Forgery



Nedir?

Server side template injection kelimelerinin kısaltması olan SSTI, saldırganların server tarafında kullanılan yerel şablonlarına kötü niyetli komutları yürütebilen girdi eklendiği bir güvenlik açığıdır. Şablon motoruna geçersiz kullanıcı girişi yerleştirildiği zaman yani giriş doğrulaması düzgün işlenmediğinde güvenlik açığı oluşur ve bu da uzaktan kod yürütülmesine (RCE) neden olabilir.

Kullanıcılar, ürünler, sonuç çıktıları hakkında bilgileri görüntülemek için kullanılan şablon motorları(template engine), dinamik verileri web sayfalarına yerleştiren sonuçların üretilmesi için şablonları bir veri modeliyle birleştirir. MVC (Model View Controller) sisteminde Model – View – Controller'dan oluşan 3 adet katman bulunur. Bu süreç kullanıcı isteği ile başlar controller → model → controller → view → çıktı ile sonlanmaktadır.

Tespiti

Kullanıcıdan alınan değer doğrudan render ediliyorsa zafiyet var demektir. Twig için örnek verecek olursak: ilgili parametreye verilen "{{7*7}}" ifadesi "49" değerini veriyorsa zafiyet var diyebiliriz.

İstismarı

Örnek payload:

GET/ssti/ssti.php?name={{_self.env.registerUndefinedFilterCallback("shell_exec")}}{{_self.env.getFilter("dir")}} HTTP/1.1

Korunma Yolları

Yapmanız gereken en iyi yol hiçbir kullanıcının yeni şablonları değiştirmesine veya göndermesine izin vermemektir.

Temizleme yapın. Herhangi bir kötü amaçlı yazılımdan kaynaklanan güvenlik açıklarını en aza indirmek için kullanıcı girişini şablonlara aktarmadan kontrol edin ve temizleyin.

Riskli karakterler kullandığınızda sandbox kullanabilirsiniz.

Kullanıcı kodunu yalnızca potansiyel olarak tehlikeli olan ve işlevlerin tamamen kaldırıldığı korumalı bir ortamda yürütün.