Yetkilendirme

Amaç

- İzin verilen işlemlerin yetki derecelerine uygun olarak sadece yetkili kullanıcılar tarafından gerçekleştirilebilmesini sağlamak
- Korunan kaynaklara erişimi, role ve yetki derecesine dayanan kararlar ile kontrol etmek
- Anonim bir kullanıcı veya sadece kayıtlı bir kullanıcı iken yönetici fonksiyonlarını kullanmak gibi hak yükseltilmesi saldırılarını önlemek.

Etkilenen Platformlar

Bütün uygulamalar.

İlgili COBIT Konuları

DS5 – Bütün bölümler gözden geçirilmeli. Bu bölüm nerdeyse COBIT'in bütün detaylı kontrol amaçlarını kapsar.

Minimum hak prensibi

Web uygulamaları genellikle, kullanıcılara veritabanı gibi korunan kaynaklar içinde lüzumsuz haklar vererek (mesela tablo düşürmek, bütün tablolarda veri çekmek veya xp_cmdshell() gibi yüksek haklı depolanmış prosedürlere izin vermek gibi), web uygulama altyapısını yüksek haklara sahip sistem hesapları ile çalıştırarak (LocalSystem veya root gibi) veya yönetilen ortamlarda (managed environments) hapishanelerinin (sandbox) dışında mutlak haklar (Java'daki AllPermission veya .NET'teki FullTrust gibi) ile koşan kodlar yazarak haddinden fazla haklar ile çalışır.

Başka bir problem bulunduğunda, haddinden fazla verilmiş haklar saldırgana sistemi veya yakınındaki başka bir sistemi daha kapsamlı ele geçirmesine yol açar. Web uygulamalarının mümkün olan en düşük haklarla çalışması gerektiğinin önemi hiç bir zaman yeterli bir şekilde belirtilemez.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

- Sistem seviyesi hesapları mümkün olduğunca düşük olmalıdırlar. “Administrator”, “root”, “sa”, “sysman”, “Supervisor”, veya başka bir diğer yüksek haklı hesap uygulamayı çalıştırmamalı, web sunucusuna, veritabanına veya orta katman ekipmanına (middleware) bağlanmamalıdır.
- Kullanıcı hesapları uygulama içinde kendilerine atanan görevleri yapmak için en düşük haklara sahip olmalıdırlar.
- Kullanıcılar yönetici olmamalıdırlar.
- Kullanıcılar yetkilendirilmemiş veya yönetimsel hiç bir fonksiyonu kullanamamalıdırlar.

Nasıl korunursunuz

- Geliştirme, test ve geçiş ortamları mümkün olabilecek en düşük yetki seviyesi ile çalışır şekilde oluşturulmalıdır ki, uygulama üretim ortamında da mümkün olabilecek en düşük yetki seviyesi ile çalışsın.
- (Çevresel sistemleri koşan) Sistem seviyesi hesapları olabilecek en düşük yetki seviyede olmalıdır. “Administrator”, “root”, “sa”, “sysman”, “Supervisor”, veya başka bir diğer yüksek haklı hesap uygulamayı çalıştırmamalı, web sunucusuna, veritabanına veya orta katman ekipmanına (middleware) bağlanmamalıdır.
- User accounts should possess just enough privileges within the application to do their assigned tasks
- Kullanıcı hesapları uygulama içinde kendilerine atanan görevleri yapmak için en düşük haklara sahip olmalıdırlar.
- Users should not be administrators and vice versa
- Ne kullanıcılar yönetici olmalı ne de tam tersi
- Users should not be able to use any unauthorized or administrative functions. See the authorization section for more details
- Kullanıcılar yetkilendirilmemiş veya yönetimsel fonksiyonları kullanamamalıdırlar. Daha fazla detay için yetkilendirme bölümüne bakınız.
- Database access should be through parameterized stored procedures (or similar) to allow all table access to be revoked (ie select, drop, update, insert,

etc) using a low privilege database account. This account should not hold any SQL roles above “user” (or similar)

- Veritabanı erişimi bütün tablo erişim haklarının (yani select, drop, update, insert, v.b.) kaldırılması için düşük haklara sahip bir veritabanı kullanıcısı kullanılarak depolanmış prosedürler yolu ile sağlanmalıdır.
- Kod erişim güvenliği değerlendirilmeli ve uygulanmalıdır. Sadece DNS isimlerine bakmanız gerekirse, kod erişimine sadece bunu yapabilmek için başvurursunuz. Bu şekilde eğer kod /etc/password’e ulaşmaya çalışırsa, erişemez ve sonlandırılır.
- Altyapı hesapları LOCAL SERVICE veya nobody kullanıcıları gibi düşük haklara sahip kullanıcılar olmalıdırlar. Ancak, eğer kod bu kullanıcılar ile koşarsa, “krallığın anahtarı” (keys to the kingdom) problemi yeniden gün yüzüne çıkar. Eğer ne yaptığınızı biliyorsanız, düşük seviyeli kullanıcılar oluşturup ve bölümlendirme için LOCAL SERVICE veya nobody gibi düşük haklı hesapların özniteliklerinin (attributes) dikkatli replikasyonu LOCAL SERVICE veya nobody paylaşmaktan daha iyidir.

Erişim Kontrol Listeleri

Bir çok erişim kontrolü kutudan güvensiz çıkar. Mesela, varsayılı Java 2 dosya sistemi politikası çoğunlukla uygulamalar tarafından gereksinim duyulmayan “All Permission”’dur.

```
grant codeBase "file:${{java.ext.dirs}}/*" {
      permission java.security.AllPermission;
};
```

Uygulamalar gereksinim duydukları minimum hakları uygulamalar ve mümkün olan en sıkı hakları zorunlu kılacak erişim kontrol listelerini oluşturmalıdırlar.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

- Herhangi bir dosya, network, kullanıcı veya başka bir sistem seviyesi erişim kontrolünün gerektiğinden fazla olduğunu bulmaya çalışın
- Kullanıcıların minimal sayıda gruba veya rollere üye olup olmadığını bulmaya çalışın
- Rollerin minimal haklara sahip olup olmadığını bulmaya çalışın
- Uygulamanın bir politika dosyası veya "zararsız mod" konfigürasyonu yolu ile minimuma indirgenmiş haklarla çalışıp çalışmadığını bulmaya çalışın

Nasıl korunursunuz

Düşünülmesi gereken erişim kontroller:

- Erişim kontrol listelerine her zaman "hepsini reddet" ile başlayın, ve sonra gerekli oldukça rolleri ve hakları ekleyin
- Network erişim kontrolleri: güvenlik duvarları ve host tabanlı filtreler
- Dosya sistemi erişim kontrolleri: dosya ve dizin hakları
- Kullanıcı erişim kontrolleri: kullanıcı ve grup platformu güvenliği

Java / .NET / PHP erişim kontrolleri: Her zaman bir Java 2 güvenlik politikası olmasını veya programsal olarak veya assembly izinleri ile .NET'te Kod Erişim güvenliğinin uygulandığını sağlayın. PHP'de dosya sistemi erişimini kısıtlamak için open_basedir direktifini içerecek şekilde "safe mode" özelliğini kullanmayı göz önünde bulundurun.

Veri erişim kontrolleri: sadece depolanmış prosedürleri kullanın ki, veritabanı kullanıcılarına verilen bir çok hakkı düşürebilesiniz – SQL enjeksiyonunu engeller.

Platformunuzu sonuna kadar (bütün güvenlik özellikleri ile) kullanmaya çalışın: Bir çok Unix tarzı işletim sisteminde erişim kontrol listelerini içeren "trusted computing base" eklentileri vardır. Windows ise bu tür eklentilerle kurulur. Bunları kullanın.

Özel yapım yetkilendirme kontrolleri

Bir çok belli başlı uygulama çatıları iyi geliştirilmiş yetkilendirme mekanizmaları içerirler (Mesela Java'nın JAAS'ı veya .NET'in web.config'de bulunan dahili yetkilendirme kabiliyetleri).

Ancak, bit çok web uygulamaları özel yapım yetkilendirme kodu içerir. Bu karmaşıklığı ve hataları arttırır. Eğer bu özellikleri kullanmamak için bir nedeniniz yoksa, kodunuz bu çatıların verdiği desteği kullanmalıdır.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

- Kodunuz çatıların size sağladığı dahili yetkilendirme kabiliyetlerini kullanıyor mu?
- Uygulama çatıların size sağladığı dahili yetkilendirme kabiliyetlerini kullanırsa daha basite indirgenebilir mi?
- Eğer özel yapım kod kullanılıyorsa, pozitif yetkilendirme ve hata durumlarını düşünün – bir kullanıcı hata oluştuğunda "yetkilendirilmiş" olabilir mi?
- Özel yapım yetkilendirme kontrolleri ile ne kadar kapsayıcı olabiliniyor? Bu mekanizma ile bütün kaynaklar ve kod korunabiliyor mu?

Nasıl korunursunuz

- Uygulamalarda her zaman az kod yazmayı tercih edin, özellikle kullanıdığınız çatı yüksek kaliteli alternatifler önerebiliyorsa.
- Eğer özel yapım kod gereksiniminiz varsa, pozitif yetkilendirme ve hata durumlarını düşünün – bir hata oluştuğunda, kullanıcının çıkmaya zorlanmasını veya en azından istenen kaynağa veya fonksiyona erişiminin engellemesini sağlayın.
- Varsayılı olarak kapsamın %100'e yaklaşmasını sağlayın.

Merkezi yetkilendirme rutinleri

Yaygın olarak yapılan bir hata, bir erişim kontrolünü bir yetkilendirme kod parçacığını kopyalayıp, yapıştırmak veya daha da kötüsü yeniden yazmaktır. İyi yazılan uygulamalar erişim kontrol rutinlerini merkezileştirirler, ki bir hata bulunduğunda tek bir yerde düzeltilirler ve heryere uygulanırlar.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

Bu saldırıya açık olan uygulamalar kodu her yerinde yetkilendirme kodu olan uygulamalardır.

Nasıl korunursunuz

- Yetkilendirme kontrollerini içeren bir kütüphane oluşturun
- Bir veya daha fazla yetkilendirme kontrollerini çağırmayı standard hale getirin

Yetkilendirme matrisi

Erişim kontrolü olan uygulamalar kullanıcıların bir sayfayı görüp göremeyeceğini veya kullanıcıların bir aksiyon gerçekleştirmeden önce buna hakkı olup olmadığını kontrol etmeliler.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

Aşağıdakileri kontrol edin:

- Bütün anonim olmayan girdi noktaları erişim kontrollerine sahip midir?
- Kontroller, aktivite üzerinde mi yoksa üstünde bir yerlerde mi?

Nasıl korunursunuz

Çatının dahili yetkilendirme kontrollerini kullanın veya merkezi bir yetkilendirme kontrolüne yapılan bir çağırıyı aksiyonun en üstüne koyun.

İstemci taraflı yetkilendirme parçaları (tokens)

Bir çok web uygulama geliştiricisi oturum deposunu kullanmaktadırlar – ki bu erişim kontrol ve gizlilik konularında yanlıştır. Bu nedenle istemci durumunu ya cookielerde, (http) başlıklarında veya gizli form alanlarında tutmaya başladılar.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

Uygulamanızı aşağıdakilere göre kontrol edin:

- İstemci taraflı yetkilendirme veya başlıklarda yetkilendirme parçacıklarını cookielerde, gizli alan formlarda veya URL argumanlarında kullanmayın.
- Herhangi bir istemci taraflı yetkilendirmeye veya yetkilendirme paçacıklarına (genelde eski kodlarda bulunurlar) güvenmeyin.

Eğer uygulamanız IBM'in Tivoli Access Manager, Netegrity'in SiteMinder veya RSA'in ClearTrust gibi SSO (tek giriş) ajanı kullanıyorsa, uygulamanızın sadece tokenları kabul etmeyip aynı zamanda denetlediğini ve bu tokenların son kullanıcıya hiç bir şekilde (başlık, cookie, gizli alanlar) görünür olmadığından emin olun. Eğer bu tokenlar son kullanıcılara gürünür şekildeyse, bütün içeriğin kriptoğrafik olarak güvenli olduğundan emin olun.

Nasıl korunursunuz

Uygulamanız bir kullanıcının kimliğinden emin olduğunda, oturum durumunu bir oturum ID ile bağdaştırın. Mesela, kullanıcı giriş yapar yapmaz, yetkilendirme derecesi ile alakalı bir bayrağı oturum nesnesinde bulunur.

```
Java
if ( authenticated ) {
}
.NET (C#)
if ( authenticated ) {

}

PHP
if ( authenticated ) {
      $_SESSION['authlevel'] = X_USER;     // X_USER başka bir yerde
yetkilendirilmiş kullanıcı anlamında tanımlanmıştır
}
```

Uygulamanızı aşağıdakilere gore control edin;

- Başlıklarda, cookielerde, gizli form alanlarında veya URL parametrelerinde hiç bir istemci taraflı kimlik doğrulama veya yetkilendirme tokenları kullanmayın.
- Hiç bir istemci taraflı kimlik doğrulama veya yetkilendirme tokenlarına (genellikle eski uygulamalarda görülür) güvenmeyin.

Eğer uygulamanız IBM'in Tivoli Access Manager, Netegrity'in SiteMinder veya RSA'in ClearTrust gibi SSO (tek giriş) ajanı kullanıyorsa, uygulamanızın sadece tokenları kabul etmeyip aynı zamanda denetlediğini ve bu tokenların son kullanıcıya hiç bir şekilde (başlık, cookie, gizli alanlar) görünür olmadığından emin olun. Eğer bu tokenlar son kullanıcılara gürünür şekildeyse, bütün içeriğin kriptoğrafik olarak güvenli olduğundan emin olun.

Korunan kaynaklara erişimi control etme

Bir çok uygulama kullanıcıların belli aksiyonları gerçekleştirip gerçekleştiremediklerini kontrol ederler, ama bundan sonra istediğini kaynağa erişimi kontrol etmezler. Mesela, forum yazılımı bir önceki mesaja cevap yazıp yazamayacağınızı kontrol eder ama istenilen mesajın korunan bir alanda mı veya gizli bir forumda mı olacağını kontrol etmez. Veya bir Internet bankacılığı uygulaması parayı transfer edip edemeyeceğinizi kontrol eder ama "hangi hesaptan" kısmındaki hesabın sizing olup olmadığını kontrol etmez.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

- Uygulama kendisinden istenen bütün kaynakların kullanıcı tarafından erişilip erişilmediğini kontrol ediyor mu?
- Dinamik sql sorguları gibi, kaynakları direk kullanan kodlar çoğunlukla model-view-controller gibi paradigmaları kullanan kodlara nazaran daha fazla risk altındadırlar. Bunun nedeni, model korunan kaynaklara erişimde kapı görevi görecek doğru noktadır ancak dinamik SQL sorguları kaynaklar hakkında genellikle yanlış kabullenmeler yaparlar.

Nasıl korunursunuz

- Korunan kaynaklara doğrudan erişmek yerine bir model kodu kullanın.
- Model kodun giriş yapmış kullanıcının kaynağa erişim hakkını kontrol etmesini sağlayın.
- Kaynağa erişimi sorgulayan kodun yeterli hata yönetimine sahip olduğunu sağlayın ve iznin her zaman verileceğini öngörmeyin.

Statik kaynaklara erişimi korumak

Bazı uygulamalar statik içerik (mesela bir işlem raporunun PDF formatı) üretirler ve altta çalışan web sunucunun bu dosyalara erişimi için sorumlu olmasına izin verirler. Çoğunlukla bu gizli bir raporun yetkilendirilmemiş erişime açık olacağı anlamına gelir.

Saldırıya açık olup olmadığınızı nasıl anlarsınız

- Uygulamanız statik içerik üretiyor veya statik içeriğe erişime izin veriyor mu?
- Statik içerik erişimi giriş yapmış kullanıcı kullanılarak mı yapılıyor?
- Eğer yapılmıyorsa, anonim bir kullanıcı korunan dokümana erişebiliyor mu?

Nasıl korunursunuz

- En iyi yöntem – statik içeriği gerektiği anda oluşturun ve web sunucunun dosya sistemine kaydetmektense direk tarayıcıya yollayın.
- Eğer statik hassas içeriği koruyorsanız, anonim erişimi engellemek için yetkilendirme kontrollerini gerçekleyin.
- Eğer diske kaydetmeniz gerekiyorsa (önerilmez), rasgele dosya isimleri (mesela bir GUID) kullanın ve geçici dosyaları düzenli olarak silin.

İleri Okuma

- ASP.NET yetkilendirme: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconaspnetauthorization.asp

Dökümanın Aslı : OWASP GUIDE 2.0.1

Çeviri : Bedirhan Urgun - urgunb@hotmail.com