Güvenli Kodlama İlkeleri

Genel Bakış

Tasarımla güvenli uygulamalar üretebilmek için yönlendirmeye ihtiyaç duyan mimarlar ve çözüm üreticiler, bunu gerçekleştirebilmek adına ana metinde belgelenen temel kontrolleri uygulamakla birlikte bu ilkelerin altında yatan "Neden?" sorusuna da tekrar gönderme yaparlar. Gizlilik, bütünlük ve kullanılabilirlik gibi güvenlik ilkeleri, önemli, geniş ve belirsiz olsalar da, değişime uğramazlar. Bu ilkeler ne kadar çok kullanılırsa, uygulamalar o denli sağlam olacaktır.

Örneğin, girilen bütün bilgi türleri için, merkezileştirilmiş bir geçerlilik rutini kapsamaya yönelik veri geçerliliği uygulamak hassas bir süreçtir. Ancak, kullanıcının gireceği tüm bilgiler için uygun hata yönetimi ve sağlam erişim kontrolüyle birlikte her dizide geçerlilik görmek çok daha hassas bir süreçtir.

Geçtiğimiz yıl, terminoloji ve sınıflandırmayı standartlaştırmaya yönelik önemli bir adım atılmıştı. Kılavuzun bu versiyonu, ilk yayınlanışında bulunan bir ya da iki ilkeyi çıkartarak temel endüstri metinlerindeki ilkelerle birlikte bütün ilkelerini iyileştirmiştir. Bu işlem, karışıklığı engellemek ve temel ilkeler bütününe uyumu artırmak amacıyla gerçekleştirilmiştir. Çıkartılan ilkeler kontroller doğrultusunda uygun bir şekilde metin içerisinde dahil edilmiştir.

Değer Sınıflandırması

Kontrollerin seçimi, ancak korunacak verinin sınıflandırılmasından sonra mümkün olabilmektedir. Örneğin, blog ve forumlar gibi düşük değerli sistemlerde uygulanabilen kontroller muhasebe, yüksek değerli bankacılık ve elektronik ticaret sistemlerine uygun kontrollerden seviye ve nicelik açısından farklıdır.

Saldırganlar hakkında

Uygulamalarınızın yanlış kullanımını engellemek üzere kontroller tasarlarken, olası saldırıları (olasılık ve gerçek kayıpları çoktan en aza indirebilmek adına) dikkate almanız gerekmektedir:

- Memnun olmayan çalışanlar ya da üreticiler
- Yan etkiler ya da virüs, solucan ve Trojan saldırılarının doğrudan sonuçları gibi "sürücü" saldırıları
- Organize suç benzeri, güdülenmiş ve suçlu saldırganlar
- Tahrif ediciler gibi organizasyonunuzun aleyhinde bulunan ancak motive olmamış suçlu saldırganlar
- Niteliksiz saldırılar

"Bilgisayar korsanlığı" terimi için hiçbir madde olmadığına dikkat çekmek isteriz. Bu, medyanın "bilgisyar korsanlığı" kelimesini duygusal ve yanlış kullanımından kaynaklanmaktadır. Doğru terim "suçlu"dur. Polise giderek daha ciddi suçluları cezalandırmaları için onlara destek olma konusunda istekli olmayan organizasyonlardan dolayı, polis tarafından yakalanan ve dava edilen tipik suçlular niteliksiz saldırganlardır.

Ancak, "bilgisyar korsanlığı" kelimesinin yanlış kullanımını düzeltmek ve kelimeyi doğru köküne döndermek için artık çok geç. Kılavuz, belirli bir özelliği aktif halde kötüye kullanma girişiminde bulunan bir şey ya da birini ifade ederken devamlı olarak "saldırgan" kelimesini kullanmaktadır.

Bilgi güvenliğinin temel dayanakları

Bilgi güvenliği aşağıdaki unsurlara dayanmaktadır:

- Gizlilik – sadece kullanıcıya izin verilen bilgiye erişim sağlama
- Bütünlük – verilerin yetkisi olmayan kullanıcılar tarafından tahrif edilmemesini ve değiştirilmemesini sağlamak
- Kullanılabilirlik – ihtiyaç duyulması halinde yetkisi olan kullanıcıların sistem ve verileri kullanabilmesi

Aşağıdaki ilkeler ise bu üç dayanakla bağlantılıdır. Aslında, kontrolün nasıl oluşturulacağına karar verme aşamasında her dayanağı da dikkate almak, sağlam bir güvenli kontrolü oluşturulmasını destekleyecektir.

Güvenlik Mimarisi

Güvenlik mimarisi bulunmayan uygulamalar sınırlı eleman analizi ve rüzgar tüneli testi yapılmadan inşa edilen köprüler gibidir. Kesinlikle bir köprüye benzerler ancak bir kelebeğin kanatlarını ilk çırpışıyla birlikte yerlebir olacaklardır. Güvenli

mimari gibi uygulama güvenliğine duyulan ihtiyaç, bina ya da köprü inşaatındaki kadar büyük ve önemlidir.

Güvenlik mimarisi temel dayanaklarla ilgilidir: uygulama, sadece doğru kullanıcılar için bilgilerin gizliliğini, verilerin bütünlüğünü korumaya ve gerektiğinde veriye erişilebilirlik sağlamaya (kullanılabilirlik) yönelik kontroller temin etmelidir. Güvenlik mimarisi, güvenlik ürünleri carnucopiasının birlikte karıştırıldığı ve "çözüm" adının verildiği "markitecture" değil, titizlikle belirlenen özellik, kontrol, daha güvenli süreçler ve aksaklık güvenlik modu bütünüdür.

Yeni bir uygulama başaltırken veya zaten var olan bir uygulamayı yeniden yapılandırırken her işlevsel özellik göz önüne alınmalı ve şunlara dikkat edilmelidir:

- Bu özellik kapsamındaki süreç olabildiğince güvenli mi? Başka bir deyişle, hatasız bir süreç mi?
- Eğer ben kötü amaçlı olsaydım, bu özelliği nasıl kötüye kullanırdım?
- Aksaklık durumunda bu özelliğin çalışır durumda olması gerekmekte midir? Eğer gerekliyse, bu özellikten kaynaklanan riskleri aza indirgemeyi sağlayacak limit ya da seçenekler mevcut mu?

Andrew van der Stock yukarıda açıklanan sürece "Kötü Düşünme™" adını veriyor, ayrıca kendinizi saldırganın yerine koyarak her türlü özelliği kötüye kullanabileceğinizi, bütün olası yöntemleri düşünmenizi, sonrasında da üç temel dayanağ dikkate alarak Stride modelini kullanmanızı tavsiye ediyor.

Bu kılavuzu takip ederek, ayrıca hem burada hem de Howard ve LeBlanc'ın kitabında açıklanan Stride/Dread tehdidi risk modelini kullanarak uygulamalarınız için resmi olarak bir güvenlik mimarisi oluşturmak için doğru yönde ilerliyor olacaksınız.

En iyi sistem mimari tasarımları ve ayrıntılı tasarım belgeleri, her özellik için risklerin nasıl azaltılacağına ve kodlama sırasında ne yapıldığına dair güvenlik bilgileri içerir.

Güvenlik mimarları iş gerekliliklerinin modellenmeye başlamasıyla güne başlarlar ve uygulamanızın son kopyası tamamlanmadan da işlerini bırakmazlar. Güvenlik tek seferlik bir kaza değil, yaşam boyu devam eden bir süreçtir.

Güvenlik İlkeleri

Bu güvenlik ilkeleri OWASP Kılavuzunun önceki versiyonundan alınmıştır ve Howard ile Leblanc'ın son derece başarılı *Writing Secure Code* (Güvenli Kod Yazımı) adlı çalışmalarında açıklanan güvenlik ilkeleri doğrultusunda düzenlenmiştir.

Saldırı Yüzey Alanını Küçültmek

Bir uygulamaya eklenen her özellik, bütün uygulamalar için belirli oranda risk teşkil eder. Güvenli gelişimin amacı saldırı yüzey alanını daraltarak genel risk oranını azaltmaktır.

Örneğin, bir web uygulaması arama işlevli online hizmet uygulamaktadır. Arama işlevi SQL enjeksiyon saldırılarına karşı korumasız olabilir. Eğer yardım özelliği yalnızca yetkisi bulunan kullanıcılarla sınırlandırılmış ise, saldırı olasılığı azaltılmış demektir. Eğer yardım özelliği merkezileştirilmiş veri geçerlilik kuralları aracılığıyla açılmışsa, SQL enjeksiyonun gerçekleştirebilme kapasitesi oldukça düşürülmüş demektir. Ancak, yardım özelliği arama özelliğini ortadan kaldırma (örneğin daha iyi kullanıcı ara alanı aracılığıyla) amacıyla yeniden yazılmış ise, yardım özelliği İnternet üzerinden büyük ölçüde erişilebilir olsa bile bu işlem saldırı yüzey alanını hemen hemen ortadan kaldıracak demektir.

Güvenli Öntanımlar

Kullanıcılara "sıradışı" bir deneyim yaşatmanın pek çok yolu vardır. Ancak, bu deneyimin öntanımla güvenli olması sağlanmalıdır. Ayrıca, eğer yetkileri var ise, güvenliklerini azaltmak kullanıcıların kararına bırakılmalıdır.

Örneğin, şifre eskitme ve karmaşıklığın öntanımla yapılabilmesi sağlanmalıdır. Uygulamalarını kullanma süreçlerini basitleştirme ve risk oranlarını artırma konusunda yukarıda adı geçen iki özelliği uygulama dışı bırakmak için kullanıcılara yetki verilebilir.

En Az Öncelik İlkesi

En az öncelik ilkesi, hesapların iş süreçlerini gerçekleştirmek en düşük oranda öncelik kapsaması gerektiğini önermektedir. Bu işlem, kullanıcı hakları, CPU sınırlandırmaları gibi kaynak izinleri, bellek, ağ ve dosya sistem izinlerini kapsar.

Örneğin, bir aracı yazılım sunucusu, sadece ağ erişimi, veritabanı tablosunu okuma erişimi ve kayıt yazma kapasitesi gerektiriyorsa, verilmesi gereken tüm izinleri tanımlıyordur. Hiçbir koşulda bu sunucuya yönetimsel öncelikler tanınmamalıdır.

Kademeli Savunma İlkesi

Kademelei savunma ilkesi, bir kontrolün akılcı olduğu durumda, riske farklı biçimlerde yaklaşan daha fazla sayıda kontrolün daha iyi olduğunu ifade eder. Kademeli olarak kullanıldıklarında kontroller, ciddi savunmasızlık durumlarınının kötüye kullanılmasını oldukça yüksek seviyede güçleştirir ve böylece oluşmalarını da engeller.

Güvenli kodlama ile, bu işlem dizi odaklı geçerlilik denetimi, merkezileştirilmiş denetim kontrolleri halini alabilir ve kullanıcıları bütün sayfalarda oturum açmak zorunda bırakabilir.

Örneğin, üretim yönetim ağlarına erişimi doğru bir şekilde sağlıyor, yönetsel kullanıcı yetkilendirmesini kotrol ediyor ve bütün erişimi kaydediyorsa, hatalı bir yönetimsel arayüzün isimsiz saldırılara karşı savunmasız kalması olası değildir

Güvenli Aksama

Uygulamalar pek çok sebepe bağlı olarak işlemleri gerçekleştiremeyebilirler. Bu aksaklığın nasıl ortaya çıktığı, uygulamanın güvenli olup olmadığını belirleyebilir.

Örneğin;

```
isAdmin = true;
try {
      codeWhichMayFail();
      isAdmin = isUserInRole( “Administrator” );
}
catch (Exception ex) {
            log.write(ex.toString());
}
```

Eğer WhichMayFail() kodunda bir aksaklık ortaya çıkarsa, kullanıcı öntanımla bir admin olur. Bu, açık bir şekilde bir güvenlik sorunudur.

Dış Sistemlerin Güvenli Olmaması

Çok sayıda organizasyon, sizlerden farklı güvenlik sistem ve tutumları uygulayan üçüncü taraf ortakların işlem yapma kapasitelerini kullanmaktadırlar. Ana

kullanıcılar, esas tedarikçiler ya da ortaklar olsalar da, dışarıdan üçüncü bir şahsı etkilemeniz ya da kontrol etmeniz mümkün değildir.

Bu nedenle, dışarı kaynaklı yürütme sistemlerine güven garantisi verilemez. Bütün dış kaynaklı sistemler benzer bir tutumla dikkate alınmalıdır.

Örneğin, bağlı bir program tedarikçisi İnternet Bankacılığı tarafından kullanılan bir veri sunmaktadır. Bunun yanında, ödül puanlarını ve potansiyel geri alım kalemlerine yönelik küçük bir liste vermektedir. Ancak, bu verilerin en son kullanıcılara görüntülenmesinin güvenli olmasını sağlamak adına kontrol edilmesi, ödül puanlarının pozitif bir sayı olduğu ve büyük olmalarının imkansız olduğundan emin olunması gerekmektedir.

Görevlerin Ayrımı

Görevlerin ayrımı, dolandırıcılığa karşı kullanılabilecek temel bir kontrol aracıdır. Örneğin, bir bilgisayar isteyen biri aynı zamanda kayıt olamaz ya da bilgisayarı doğrudan alamaz. Böylece, kullanıcının çok sayıda bilgisayar istemesi ve eline ulaşmadığını iddia etmesi engellenmiş olacaktır.

Belirli görevlerin, normal kullanıcılara göre farklı güven seviyeleri vardır. Özellikle, yöneticiler normal kullanıcılardan farklıdır. Genelde, yöneticiler uygulama kullanıcıları olmamalıdırlar.

Örneğin, bir yönetici sistemi açabilmeli ya da kapatilmeli, şifre politikasını belirleyebilmeli ancak süper öncelikli bir kullanıcı gibi depo bölümünde oturum açıp diğer kullanıcılar adına eşya “satın” alamamalıdır.

Belirsizlik Yolu ile Güvenliğe Güvenmeyin

Belirsizlik yolu ile güvenlik zayıf bir güvenlik kontrolüdür. Tek kontrol aracı olduğu pek çok durumda da hemen hemen her zaman başarısız olur. Sır tutmanın kötü bir fikir olduğu anlamına gelmemekle birlikte, kilit sistemlerin güvenliğinin ayrıntıların gizli tutulması esasına dayandırılmaması ifade edilmektedir.

Örneğin, bir uygulamanın güvenliği, gizlenen bir kaynak kod bilgisine dayandırılmamalıdır. Güvenlik pek çok etmene dayandırılmalıdır. Akla yatkın şifre politikaları, kademeli savunma, iş bağlamında işlem sınırlamaları, sağlam ağ mimarisi, dolandırıcılık ve denetim kontrolleri de dikkate alınmalıdır.

Linux, oldukça pratik bir örnektir. Linux kaynak kodu geniş ölçüde erişilebilirdir. Bununla birlikte doğru bir şekilde güvence altına alındığında, zorlu, güvenli ve sağlam bir işlem sistemidir.

Basitlik

Saldırı yüzey alanı ve basitlik yakından ilişkilidir. Belirli yazılım mühendisliği akımları, diğer bir şekilde doğrudan ve basit olabilecek kodlara yönelik oldukça karmaşık yaklaşımlar izlemektedirler.

Daha basit yaklaşımların daha hızlı ve basit olacağı durumlarda, sistem geliştiricilerin de çifte negatif ve karmaşık mimari kullanımında kaçınmaları gerekmektedir.

Örneğin, ayrı bir aracı yazılım sunucusunda çok sayıda tek varlık kullanımı çok moda olsa da, yarış durumlarına karşı korunma amacıyla uygun mutex mekanizmasıyla birlikte küresel değişkenler kullanmak hem daha güvenli hem de daha hızlıdır.

Güvenlik Sorunlarını Uygun Şekilde Ele Alın

Bir güvenlik sorunu belirlendiğinde, sorun için bir test süreci oluşturmak ve soruna yol açan esas meselenin anlaşılması önemlidir. Tasarım biçimleri kullanıldığında, güvenli sorununun bütün kod tabanlarına yayılmış olması olasıdır. Bu nedenle gerilemelere yol açmadan doğru çözüm yolunu geliştirmek esastır.

Örneğin, bir kullanıcı, cookielerini uyumlaştırarak diğer bir kullanıcının bakiyesini görebilmektedir. Direk çözüm yolu ortadadır, ancak cookie kullanım kodu bütün uygulamalalar tarafından paylaşılmaktadır, bu nedenle yalnızca bir uygulama züerinde yapılacak değişiklik bütün diğer uygulamalrı etkileyecektir. Çözüm yolu, etkilenen bütün uygulamalar üzerinde denenmelidir.

Dökümanın Aslı : OWASP GUIDE 2.0.1

Çeviri : Çiğdem Akanyıldız - acigdema@yahoo.com