4.3 İŞ MANTIĞININ TESTİ

KISA ÖZET

İş mantığı şunlardan oluşmaktadır:

- İş politikalarını (kanallar, yerleşim, lojistik, fiyatlar ve ürünler gibi) ifade eden iş kuralları ve
- Bir katılımcıdan (bu bir kişi veya yazılım sistemi olabilir) başka bir katılımcıya sıralanmış görevlerin veri veya belgelerini ifade eden iş akışları

Bir uygulamanın iş mantığına yönelik saldırılar tehlikelidir, zor belirlenir ve uygulamata özel olarak yapılır.

DESCRIPTION OF THE ISSUE

İş mantığı, iş kuralları tarafından izin verilmeyen işlemlerin kullanıcı tarafından yapılabilmesine olanak sağlayacak gedikleri barındırabilir. Örneğin, eğer geri ödemeler üzerinden 1000$'dan fazla bir limit varsa, saldırganın sistemdem izin verilenin üzerinde para çekmece şekilde suistimal etmesi mümkün müdür? Veya belki de işlemleri yapmak konusunda bir sıra öngörmüş olabilirsiniz, ancak saldırgan sıralamayı gözardı edecek girişimde bulunabilir. Veya herhangibir kullanıcı eksi miktarda bir parayla satınalma yapabiliyor mu? Sıklıkla uygulamalarda bu tip iş mantığı güvenlik kontroleri bulunmamaktadır.

Hazır araçlar ise içeriği anlamakta zorlanmaktadır, sonucunda bu tip testlerin yapılması kişiye kalmaktadır.

İş Sınırlamaları ve Kısıtları

Uygulama tarafından sağlanan işlevlerin kuralları dikkate alınmalıdır. Kullanıcı hareketlerinde sınırlama veya kısıtlama var mı? Ardından uygulamanın bu kurallar konusunda zorlamasının olup olmadığı düşünülmelidir. Eğer yapılan işe aşinaysanız, genellikle uygulamayı doğrulamak için test ve analizleri belirlemek oldukça kolay olacaktır. Eğer üçüncü parti bir test görevlisiyseniz, bu durumda sağduyunuzla uygulamanın iş için farklı işlemlere izin verip vermediğini sorgulamanız gerekir.

Örnek: Bir e-ticaret sitesinde ürün sayısını eksi bir sayı olarak belirlenmesi sonucunda saldırgana para yatırılabilir. Bu soruna önlem olarak uygulamada, alışveriş kartındaki miktar alanlarına eksi sayıların girilmesine izin vermeyecek şekilde daha güçlü veri doğrulamaların kullanılması gerekir.

KARA KUTU TESTİ VE ÖRNEKLER

Her ne kadar mantıksal zaafiyetleri ortaya çıkarmak her zaman için bir sanat olarak kalacak olsa da, sistematik olarak üzerine gidilirse önemli bir mesafe katedilebilir. Önerilen yaklaşımın şunları içermelidir:

- Uygulamayı anlama
- Mantıksal testleri tasarlamak için ham veri yaratma
- Mantıksal testleri tasarlama
- Standart öngerekilikler
- Mantıksal testleri çalıştırma

Uygulamayı anlamak

Uygulamayı baştan aşağı anlamak, maktıksal testleri tasarlamak için öngerekliliktir. Başlamak için:

- Uygulamanın işlevlerini anlatan tüm belgeleri elde edin. Örneğin:
    - Uygulama kitapçıkları
    - Gerekli belgeler
    - İşlevsel özellikler
    - Kullanım ve/veya Suistimal Senaryoları
- Uygulamayı keşfetme, ve onaylanan kullanım senaryoları ile izin verilen haklar içinde değişik kullanıcılarla uygulamanın tüm değişik kullanım yollarını anlamaya çalışma

Mantıksal testleri tasarlamak için ham veri yaratma

İdeal olarak bu aşama aşağıdakiler gündeme gelecektir:

- Uygulamanın tüm **iş senaryoları**. Örneğin; bir e-ticaret uygulaması için şunlara benzerebilir,
    - Ürün siparişi
    - Ödeme
    - Tarama
    - Ürün arama
- **İş akışları**. İş senaryolarından farklı olarak iş akışları değişik kullanıcıları da içermektedir. Örneklerin içeriği:
    - Sipariş oluşturma ve onaylama
    - İlan tahtası (bir kullanıcının gönderdiği yazı, moderatör tarafından incelenir ve sonuçta diğer kullanıcılar tarafından görülebilir)
- Farklı **kullanıcı rolleri**
    - Sistem Yöneticisi
    - Müdür
    - Çalışan
    - Yönetim Kurulu Başkanı (CEO)
- Farklı **gruplar veya birimler** (ağaç yapısında olabilir (örneğin mühendislik biriminin altındaki satış grubu gibi) veya etiket görünümünde üyelikleri olabilir (örneğin biri hem satış hem de pazarlama grubu üyesi olabilir)
    - Satınalma

- Pazarlama
- Mühendislik

- **Değişik kullanıcı v grupların erişim hakları** – Uygulama değişik hakara sahip kullanıcıların bazı kaynak (veya varlıklar) üzerindeki yetkilerine izin verir ve bu yetki sınırlarının belirlenmesi gerekir. Uygulama belgelerinden etkili biçimde yararlanmak, iş kuralları ve sınırlamaların öğrenilmesi için kolay bir yoldur. Örneğin; "Eğer sistem yöneticisi bireysel kullanıcı erişimine izin verdiyse...." veya "Eğer sistem yöneticisi tarafından kurgulandıysa... " gibi ifadelerin varlığına bakıp uygulamaya yüklenen kısıtlamaları öğrenebilirsiniz.

- **Yetki Çizelgesi** – Kaynaklar üzerindeki kısıtlamalarla verilen değişik yetkileri öğrendikten sonra, Yetki Çizelgesini yaratabilirsiniz. Şu sorulara yanıtlar gerekmektedir:

  - Hangi kaynak üzerinde hangi kısıtlamayla kullanıcı roleri neler yapabilmektedir? Bu size kimin hangi kaynak üzerinde ne yapamayacağı konusunun ortaya çıkmasında yardımcı olacaktır.
  - Gruplar arasındaki politikalar nelerdir?

Şu yetkileri dikkate alın: "Harcama raporunun onaylanması", "Toplantı odasının ayırılması", "Bir hesaptan diğerine para aktarılması". Yetkiler bir fiil (örneğin; onaylama, iptal etme, aktarılma) ve bir isim veya isimlerden oluşan tamlamadan (örneğin; harcama raporu, toplantı odası, hesap) oluşan bir birleşim olarak düşünülebilir. Bu çalışmanın çıktısı olarak, değişik yetkilerin sıralandığı ve bu yetkilere karşılık gelen kullanıcı rollerinden ve gruplarından oluşturan bir tablo elde edilir. Ayrıca bu tablodaki verilerin niteliği eklenecek bir "Yorumlar" sütunuyla artırılabilir.

Yetki	**Kim yapabilir**	**Yorumlar**
Harcama raporu onayı	Herhangi bir grup yöneticisi astlarının raporlarını onaylayabilir	
Harcama raporunun ibraz edilemesi	Herhangi bir çalışan kendisi için yapabilir	
hesaplar arası kaynak aktarımı	Hesap sahibi kendi hesabından başka bir hesaba aktarım yapabilir	
Ödeme makbuzunun görülmesi	Herhangibir çalışan kendisine ait olanları görebilir	

Bu veri mantıksal testler için anahtar bir girdidir.

Mantıksal testleri tasarlama

Ham verileri yığınından yola çıkarak mantıksal testleri tasarlamak için oluşturulan çeşitli yönergeler:

- **Yetki Tablosu** – Uygulamaya özel mantıksal tehditleri belirlerken kaynak olarak yetki tablosundan yararlanılmalıdır. Genel olarak, tüm sistem yöneticisi yetkileri için bir test geliştirip çok yetkiye sahip veya yetkisiz bir kullanıcı rolüyle izinsiz olarak çalıştırılıp çalıştırılmadığı kontrol edilir. Örneğin:
    - Yetki: İşletme Müdürü müşteri siparişini onaylayamaz
    - Mantıksal Test: İşletme Müdürü müşteri siparişini onaylar
- **Özel kullanıcı hareket silsilelerinin uygunsuz kullanımı –** Uygulamadaki sayfalarda belirli bir yoldan dolaşmak veya eşzamansız tekrar ziyaret etmek, uygulamanın yapması gerekenden farklı anlamda işler yapacak mantıksal hatalara neden olabilir. Örneğin:
    - Kullanıcının dolduracağı ve sonraki basamağa geçeceği bir uygulama sihirbazını elealandığında; (geliştiricilere göre) kullanıcı normal yolların sihirbazın işlemlerinin ortasından giriş yapamaz. Ortadaki bir basamağın (diyelim ki 7 basamaklı bir işlemde 4. basamağın) yer imi olarak saklanması, sonra diğer basamakların sonuna kadar veya formun gönderimine kadar sürdürülmesi ve sonra daha önce yer imi olarak kayıt edilen ortadaki basamağın tekrar ziyaret edilmesi, geri plandaki mantık *zayıf modelleme*den kanaklı olarak kırılabilir.
- **Bütün iş görme yollarının kapsanması** – Testleri tasarlarken, aynı işi yerine getirmek için bütün değişik yolların kontrol edin. Örneğin; hem nakit hem de kredi ödeme işlemleri için testleri yaratın.
- **Kullanıcı tarafı doğrulama** – Kullanıcı tarafındaki bütün doğrulamalara bakın ve mantıksal testlerin tasarımında nasıl bir temel oluşturacağını görürsünüz. Örneğin; bir para aktarımı işlemi sırasında eksi değerler için miktar alanı bir doğrulamaya sahiptir. Bu bilgi, "kullanıcının eksi miktarda para aktarımı" şeklinde bir mantıksal test tasarlanmasında kullanılabilir.

Standart öngereklilikler

Bazı ön işlemler, kurulum gibi düşünülür

- Değişik yetkilere sahip test kullanıcıları yaratın
- Uygulama içindeki önemli bütün iş senaryoları ve iş akışlarını tarayın

Mantıksal testlerin çalıştırılması

Her mantıksal testi seçin ve aşağıdakileri yapın:

- Mantıksal testi karşılayan kabul edilebilir kullanım senaryosunda öncelikli olarak HTTP/S isteklerini inceleyin
    - HTTP/S isteklerinin sırasını kontrol edin
    - Aktarılan dizi sorgu parametreleri, form alanları ve gizli alanların amacını anlayın,
- Bilenen zaafiyetlerle yararlanıp bozmayı deneyin
- Test için uygulamanın çöktüğünü soruşturun

REFERANSLAR

Makaleler

- İş Mantığı - http://en.wikipedia.org/wiki/Business_logic
- Ses uygulama güvenliği örnekleriyle uygulama mantığı saldırılarının önlenmesi http://searchappsecurity.techtarget.com/qna/0,289202,sid92_gci1213424,00.html?bucket=NEWS&topic=302570

Araçlar

- Hazır araçlar mantıksal zaafiyetleri belirlemede yeteneksizdir. Örneğin; bu tip araçlar için, bir bankanın "para aktarımı" sayfasının bir kullanıcının başka bir kullanıcıya eksi miktarda aktarımına izin vermesi (yani bir kullanıcının artı değerde bir miktarı kendi hesabına aktarmasına izin veriliyor) anlamsızdır veya araçlar test görevlilerini bu durumdan kuşkulandırabilecek mekanizmalara sahip değildir.

Eksi bir miktarın aktarımını engelleme: Kullanıcı tarafındaki doğrulamaları test edebilecek biçimde araçlar genişletilebilir. Örneğin; araç, bir formu garip değerlerle doldurma ve bunu tarayıcı özelliklerini kullanarak göndermeyi deneme özelliğine sahip olabilir. Böylece tarayıcının gerçekten isteği gönderip göndermediği kontrol edilebilir. Tarayıcının isteği göndermemesinin belirlenmesi, araca uyarı olarak gönderilmiş değerlerin, kullanıcı tarafındaki doğrulamayla kabul edilmediği şeklinde gelebilir. Bunun test görevlisine rapor edilmesiyle kullanıcı tarafındaki doğrulamayı atlatabilecek mantıksal testlerin tasarımının gerekiliği anlaşılır. "Eksi para aktarımı" örneğinde olduğu gibi, test görevlisi için böyle bir transfer ilginç bir test olabilir. Test görevlisi, kullanıcı tarafı doğrulamasını atlatacak koda sahip testi tasarlayabilmeli ve "para aktarımı başarılı oldu" yanıtının geri dönüp dönmediğini kontrol edebilmelidir. Dikkat edilmesi gereken nokta aracın bu ve benzeri zaafiyetleri belirlemesi değil ama test yapanların benzer mantıksal zaafiyetleri bulunmasında yardımcı olacak bu tip işlevlerin eklenebilmesinin mümkün olabilmesidir.

4.4 KİMLİK DENETİMİ TESTLERİ

Kimlik denetimi ("Authentication", Yunanca: αυθεντικός = sahici veya gerçek, 'authentes' = yazar) birşeyin (veya birinin) yetkisinin tanıtılması veya doğrulanmasıdır. Bir nesnenin kimlik denetimi kaynağının belirlenmesi anlamına gelirken, genellikle bir kişinin kimlik denetimi kimliğin doğruluğunun soruşturulmasını içerir. Kimlik denetimi bir veya birden çok etmene dayanmaktadır. Bilgisayar güvenliğinde, kimlik denetimidir bir iletişimde göndericinin sayısal kimliğinin doğrulanması işlemidir. Bu tip bir işleme en yaygın örnek oturum açma işlemidir. Kimlik denetimi planının test edilmesi, kimlik doğrulama işleminin nasıl çalıştığı ve bu bilginin denetim mekanizmasının atlatılmasında nasıl kullanıldığının anlaşılması anlamına gelmektedir.

Varsayılan veya tahmin edilebilen (sözlük) kullanıcı hesabı
İlk olarak varsayılan kullanıcı hesapları veya tahmin edilebilen kullanıcı adı ve parolalar olup olmadığının (sözlüklerden) kontrolünü yaparız.

Kaba Kuvvet (Brute Force)
Sözlük tipi saldırılar başarısız olduğunda test görevlisi kimlik denetiminden geçmek için kaba kuvvet yöntemlerini kullanır. Zaman gerektirdiği için ve olası dışa atılmalar yüzünen test yapanlar için kaba kuvvet testlerini başarmak kolay değildir.

Kimlik denetimi planının atlatılması
Diğer pasif test yöntemleri, kimlik denetim planını, uygulamanın tüm kaynaklarının yeterli derecede korunmadığını algılayarak atlatmayı denemektedir. Test edenler bu kaynaklara yetkisiz erişebilmektedir.

Directory traversal/file include
Directory Traversal Testing is a particular method to find a way to bypass the application and gain access to system resources. Typically, these vulnerabilities are caused by misconfiguration.

Vulnerable remember password and pwd reset
Here we test how the application manages the process of "password forgotten". We also check whether the application allows the user to store the password in the browser ("remember password" function).

Logout and Browser Cache Management Testing
As a final test we check that the logout and caching functions are properly implemented.

Dökümanın Aslı : OWASP TESTING GUIDE v2

Çeviri : Türker Gülüm – turker.gulum@gmail.com