Mobil Ortamda Geleneksel Kullanıcı Detayları Servisi İnşası

Mirsat Yeşiltepe[1]

[1] Yıldız Teknik Üniversitesi, Matematik Mühendisliği Bölümü, İstanbul

mirsaty@yildiz.edu.tr

Özet: Bulut servisine istek gönderen mobil cihazlardaki istemcilerin kimlik doğrulama ve kendilerini hatırlatma için oturum çerezleri kullanmaları çok iyi bir yol değildir[1]. Örneğin bir sunucu düşünelim. Sunucudaki kaynaklara erişebilen bir istemci olsun. Bu istemci sunucunun sunduğu birkaç hizmetten faydalanmak isteyebilir. Bu durumun çözümü klasik kimlik doğrulama mekanizmaları yetersiz kalmaktadır. Bu çalışmada amaç bu durumu çözmek için kullanılan gelenesek kullanıcı detayları servisini mobil ortamlar için incelemektir.

Anahtar Sözcükler: İstek, kimlik doğrulama, HTTP, yetkilendirme.

Abstract: Cloud services allow clients to request sender authentication on mobile devices and use session cookies to remind themselves is not a very good way. For example, suppose a server. Whether a client can access resources on the server. This client may want to take advantage of several services offered by the server. In this case the solution of classic authentication mechanisms are inadequate . The aim of this study is used to solve this situation is to examine the gelenesek user details service for the mobile environment.

1. Giriş

Bulut servisi inşasında servisi farklı mobil cihazların kullanma isteği dikkate alınılmalıdır. Bu ortamda Android cihazlar olabileceği gibi iOS cihazlar da olabilir. Yani birbirinden farklı ortamlarda çalışan mobil cihazlar aynı sunucu ile iletişimde olmak isteyebilir. Farklı telefondaki uygulamaların aynı sunucuya ulaşmak istediğini düşünelim. Uygulamalardan biri sunucudaki bir kaynağa başka bir uygulama da sunucudaki farklı bir kaynağa erişmek istesin.

Her bir uygulamanın kendi tanımlayıcılarını sunucu ile ilişkilendirerek kendini tanıttığı ve sunucudaki servislerden nasıl faydalanıldığı bu bölümün konusudur. Eğer bağlantı sayfasında bağlanıldıktan sonra oturum çerezi kullanılarak oturum standardı kullanılmak istenirse her uygulamaya bağlantı sayfası gösterilip kullanıcı ismi ve şifresi istenecektir.

Bazı uygulamalar istemcilerin uygulamaya her bağlanmak istediği durumda kimlik doğrulama yapmasını zorunlu koşmaz. İstemcinin kullanıcı ismi ve şifresini depolar. Böyle olunca da kullanıcı ismi ve şifre bilgilerinin farklı uygulamaların her birinde saklanması durumu ortaya çıkar [2].

Bazen farklı mobil cihazlar aynı sunucu ile ilgilenirken bazen de istemci farklı uygulamalara bağlanmak isteyebilir. Her uygulama için her zaman bağlantı sayfası ile meşgul olmaması istemciler tarafından istenilen bir durum değildir.

2. Kullanıcı Detayları Ara Yüzü

Güvenlikte hiçbir zaman unutulmaması gereken nokta güvenliği en zayıf noktanın belirlediğidir. Örneğin mesaj iletişiminde TripleDes şifreleme algoritması, kimlik doğrulama da ise Des şifreleme algoritmasının kullanıldığını düşünelim. Sistemin güvenliği esasında Des ile korunuyor gibi olur.

Kötü niyetli kullanıcı da kimlik doğrulama üzerinden ortamı bozup bilgileri çalmak isteyecektir. Yani en düşük güvenlikli kısım kötü niyetli istemci için davetiye olacaktır. Burada güvenliğin sadece kriptoloji ile sağlanmadığını da unutmamak gerekir. Şifreleme bir arabadır. Pahalı araba kullanan sürücü kaza yapma ihtimali azdır. Fakat eğer sürücü arabayı etkin kullanmayı bilmezse kaza yapma ihtimalini düşmeyecektir.

Mobil dünyada da eğer uygulama bizim kimlik bilgilerimizi düz metin, basit şifreleme veya başka basit mekanizmalar ile korunursa kötü niyetliler ilk o uygulama üzerinden saldırır. Bu sebeple birden fazla uygulama için aynı kullanıcı ismi ve şifrenin kullanılması fikri iyi olmaz. Öbür türlü de sunucudaki her bir uygulama için istemci her uygulamaya bağlanmak istediğinde kimlik doğrulama mekanizmasından geçmesi gerekir. Bu durumda oturum çerezinden vazgeçilmiş olması bir dezavantajdır[3].

İki adet uygulamanın yer aldığını düşünelim. İlk uygulama çok güvenilir olduğu ikinci uygulamanın ise daha az güvenilir olsun. Eğer ikisi de aynı kimlik doğrulamayı kullanıyorsa(aynı bilgilerle) yapılacak bir şey kalmaz. Her ikisi de aynı kullanıcı ismi ve şifre ile sunucuya bağlanır ve istemcinin hesap bilgilerini kullanır.

Her uygulamanın her şeyi yapması istenmez. Belli şeyleri yapması istenebilir. Bazen de sadece olan şeyleri görüntülemek isteyen uygulamalar olabilir.

Uygulamaların normalde her hakka sahip olması istenmez. Uygulamaların gerekliliklerine göre haklara sahip olması beklenir. Çok uygulamaya tanımlayıcı bilgisi vermek uygulamaların bunları nasıl ve ne şekilde sakladığı bilinmeyen bir ortamda sağlıklı olmayacaktır.

Uygulamalara bizim sahip olduğumuz yetkilere göre erişebildiğimiz kaynakları direk(kimlik doğrulama olmadan) vermek bir çözümdür. Bu yöntem güvenilen uygulamalara ve sadece var olan bilgileri görüp üzerinde oynama yapmak istemeyen uygulamalar için uygundur. Bu model uygulamada kullanıcı ismi ve şifre ile oturum çerezi oluşturarak hesaplara erişim için uygun değildir[4].

Özetle kullanıcı ismi ve şifreyi ya her uygulama için uygulama kullanılmak istenildiğinde veya uygulama mobil cihazda bunu saklayarak kullanılır. Uygulamada bu tanımlayıcıları başka bir yerde saklar. Aksi takdirde geliştirici istemediği durumda bile istemcileri bağlantı sayfasına yönlendirir. Bu nedenle uygulamanın belirlediği başka bir yerde tanımlayıcılar olur.

Herhangi bir tanecikli yapı türünde hesaba bağlı olarak kaynaklara erişimin düzenlenmesi yoktur. Hesabın farklı kısımlarına farklı uygulamaların erişimi mobil ortamlar için uygun değildir ve bırakılmaya başlanmıştır.

Kullanıcının uygulamaya her bağlandığından kendisini kullanıcı ismi ve şifre ile tanıtmasının bazı dezavantajları vardır. Mobil modelde bu durum çok istenilmez.

Verilerimize erişim için çoklu uygulamalarla çalışıldığını düşünelim. Yani ortamda kullanıcının iletişim kurmak istediği birden fazla uygulama vardır. Baz uygulamaları birden fazla cihaza hizmet etsin. Diğer cihazlar aynı uygulamaya farklı cihaz türleri ile bağlanır. Kullanıcılar birden fazla uygulamaya bağlanır. Bir uygulama için kullanıcıların şifreleri farklı yerlerde saklanır. Uygulamalar elverişli olmak için şifreleri saklar[5]. Eğer kullanıcı şifresini değiştirmek isterse ki bu genellikler güvenlik için yapılır kendisini günceller ve şifresini değiştirir. Her zaman şifre değiştirilmek istenilsin. Bu durumda her uyulamaya girilmek istenildiğin de şifre değiştirilir. Her şifre değiştirilme isteğinde kullanıcının kendini güncellemesi kullanıcının gözünde zaman alıcı ve boş bir iş olarak gözükmeye başlar. Bu yüzden her

zaman değil ara sıra şifre değiştirmek daha uygun gözükür.

Başka bir sorun da uygulamaya bağlantı iznini verilmesi durumundaki ortamdır. Böyle olunca sunucu hangi uygulamanın erişme hakkının olduğunu bilemez. Örneğin bir uygulamanın bizim politikalarımıza bağlı olarak sadece belli şeylere erişmesi istenebilir. Bu işleme sunucu dâhil edilmez. Uygulamaya kullanıcı adı ve şifre bilgileri verilerek uygulama ortamdan gider ve sunucu ile olan iletişim başlatılır. Sunucu, uygulamanın erişim hakkının olup olmadığı ile ilgilenmez. Sunucu sadece uygulamanın kullanıcı üzerinden erişim bilgileri ile erişilip erişilmediği ile ilgilenir. Arka plan ile ilgilenmez.

Belli bir süre sonra bu uygulamaya güvenilmemeye başlanıldığında alınacak tedbir ve işlemlerde önemli bir konudur. Bir uygulamaya güvenilmeyip diğerlerine güvenmek olası bir durumdur. Bu durumda yapılabilecek şey ilk uygulamadan kullanıcı ismi ve şifre ile sağlanan bağlantı bilgilerini geri çekmektir. Sonrada kullanıcı ismi ve şifre ile oluşan tanımlayıcı bilgilerinin değiştirilerek güvenilmeyen uygulamanın kullanıcının bilgileri ile sunucuya bağlanamaması yapılacak ikinci adımdır. İşlemler yapıldığında artık güvenilen uygulama kullanıcının tanımlayıcılarını kullanarak sunucuya erişemeyecektir.

Normalde kullanıcı ismi ve şifre bilgisi oturum çerezi yaratmak için kullanılır[6]. Fakat bu model mobil dünyadaki kullanıcıların olduğu ve çoklu uygulamaların farklı veri veya veri gruplarına ulaşmak istediği bir ortamda uygun değildir. Bazı uygulamalara diğerlerinden daha çok güvenilebilir. Fakat kullanıcı tanımlayıcıları ile hangi uygulama olursa olsun sunucuya tam erişim yetkisi verilirse bulut bazlı servislerin ideal olmayan durumunu oluşturarak güvenlik problemlerini ortaya çıkarır.

3. Kullanıcı Detayları Ara Yüzü Kullanımı

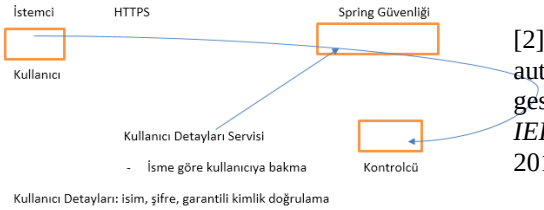
Kullanıcıya özel verileri yükleyen çekirdek ara yüzüdür. Kullanıcı DAO olarak çerçeve kullanılmaktadır ve DaoAuthenticationProvider(DAO Kimlik Doğrulama Sağlayıcı) tarafından kullanılan bir stratejidir. Ara yüzü yeni veri erişim stratejileri için destek kolaylaştırır sadece bir salt okunur yöntemi gerektirir.

Alt ara yüzünün ismi UserDetailsManager(Kullanıcı Detayları Yöneticisi)'dir[7]. Alakalı sınıfları ise CachingUserDetailsService(Kullanıcı Detaylarını Önyükleme Servisi) , InMemoryDaoImpl (Hafızada ki DOA Implementasyonu) , InMemoryUserDetailsManager(Hafızada ki Kullanıcı Detayları Yöneticisi Servisi) ,JdbcDaoImpl (JDBC DOA İmplementasyonu) ,JdbcUserDetailsManager (JDBC

Kullanıcı Detayları Yöneticisi), LdapUserDetailsManage(LDAP Kullanıcı Detayları Yöneticisi) , LdapUserDetailsService (LDAP Kullanıcı Detayları Servisi) , UserDetailsServiceWrapper (Kullanıcı Bilgileri Servis Sarıcı)'dır.

Metodun türü kullanıcı detayları olup, "loadUserByUsername(String username)" şeklinde kullanılarak kullanıcı ismine bağlı kullanıcı bilgilerinin yerini işaret eder. Asla "null" dönmez[8].

Kısaca kullanıcı adı ile eşleşen kullanıcıyı bulur. Gerçek uygulamada, arama muhtemelen uygulama örneğinin yapılandırıldığına bağlı harf duyarlı veya harf duyarsız olabilir. Bu durumda, UserDetails(Kullanıcı Detayları) aslında istenenden farklı bir durumda olan kullanıcı adı olabilen bir nesne döndürür.



Şekil 1. Kullanıcı detayları servisi kullanım şeması [9]

4. Sonuç

Kullanıcı hesabı ile ilişkili veri kaynaklarına erişimde erişimin kontrolü istenilen bir durumdur. Burada amaç kullanıcının hiçbir uygulamaya kullanıcı tanımlayıcılarını vermemesi değildir. Kullanıcı verilerine hangi uygulamanın ne kadar ulaşacağı önemli bir konudur.

Yapılmak istenen kullanıcının tanımlayıcılarını değiştirirken uygulamaların erişimlerinin bundan etkilenmemesidir. Benzer şekilde belirli bir uygulama için erişimin kaldırılabilmesidir. Bu işlemler bağlantı ekranından girilen verilerle oluşturulan oturum çerezlerinin kullanılarak uygulamalara tam erişim verildiği ortamlarda kolay değildir.

Kullanıcıların kimliğini doğrulama ve kendi oturumlarını tutma arasında sıkı bir ilişki vardır. Doğrulanmış oturumları kaçırma, tespit ve tekrar karşı korunması işleriyle uğraşılırken gereken, anonim oturumları için çok hassas değildir. Aslında, geçerli bir oturum belirteci oturumu doğrular.

Kaynaklar

[1] Shrader, Theodore Jack London, Garry L. Child, and William H. Gengler. "Dynamic use and validation of HTTP cookies for authentication." U.S. Patent No. 6,374,359. 16 Apr. 2002.

[2] Feng, Tao, et al. "Continuous mobile authentication using touchscreen gestures." *Homeland Security (HST), 2012 IEEE Conference on Technologies for*. IEEE, 2012.

[3] Embassy, U. S. "Gizlilik." (2010).

[4] Glass, Steven, et al. "Mobile IP authentication, authorization, and accounting requirements." *Work in Progress* (2000).

[5] Wang, Guojun, Qin Liu, and Jie Wu. "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services." *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010.

[6] Toprak, Özgür, and Seyhun Altunbay. "Java ve SOAP kullanılarak Mobil Cihazlardan Hisse Senedi Alımı Uygulaması."

[7] Deinum, Marten, et al. "Spring Social." *Spring Recipes*. Apress, 2014. 303-330.

[8] Scarioni, Carlo. "Customizing and Extending Spring Security." *Pro Spring Security*. Apress, 2013. 237-272.

[9] Dr. Douglas C. Schmidt, Vanderbilt University, "Programming Mobile Services for Android Handheld Systems: Communication", 2015.