Mobil ve Web Uygulamalarının Yazılım Güvenliği

Hanım Eken[1]

[1] E-Devlet ve Bilgi Toplumu Direktörlüğü, TÜRKSAT AŞ, Ankara
heken@turksat.com.tr

Özet: Dünya genelinde internette, insanların kişisel bilgisayarlarının masaüstü uygulamalarında ve mobil ortamda binlerce uygulama çalışmaktadır. Bu uygulamalar insanların işlerini kolaylaştırmaktadır. İnsanlar bu uygulamaları kullanarak bir çok işlemini kolayca yapabilmektedir. Bu uygulamaların kullanılması çok büyük boyuttaki bilgide depolanmaktadır. Bu nedenle veritabanları büyük boyutta bilgi içermektedir. Bilginin bu kadar büyük boyutlara ulaşması bilginin güvenirliği ve korunması gerekliliğini ortaya çıkarmaktadır. Ayrıca kötü niyetli insanların amacı sadece bilgiyi ele geçirmek olmayabilir. Sistemi veya uygulamayı ulaşılmaz hale getirmekte olabilir. Giderek daha fazla sayıda Web Tabanlı kurumsal uygulamalar arttığı için özellikle, web uygulamalarının güvenliği giderek daha önemli hale gelmiştir. Uygulamada kesintinin olması milyonlarca dolar zarar neden olabilmektedir. Çünkü mali ve tıbbi gibi hassas veriler bu uygulamalar tarafından kullanılmaktadır. Dolayısıyla uygulamaları saldırılara karşı korumak çok önemlidir. Bu çalışmada uygulamalarındaki tehdit ve açıklardan ve buna karşın nasıl önlemler alınabileceğinden bahsedilecektir. Bu çalışmanın, kurumsal bilgi güvenliğinin yüksek seviyede sağlanmasına yönelik farkındalık oluşturması, mevcut ve yeni standartlar hakkında daha fazla bilgi içermesi, literatür özetini sunması ve yüksek seviyede güvenliğin sağlanmasına katkılar sağlayacağı düşünülmektedir.

Anahtar Sözcükler: Bilgi Güvenliği, Bilgi Güvenliği Yönetimi, Web Uygulama Güvenliği, Mobil Yazılımı Güvenliği.

Software Security of Mobile and Web Application

Abstract: Today, thousands of applications world-wide web, and mobile media applications are used by people. People are using these applications in a multi-process, using the Internet or mobile devices can easily. These applications facilitate the lives of people. People have been entering a significant amount of information into these applications by information screens everyday. Therefore, applications and databases of these application contain a large amount of information. Each user can only have access to their own knowledge. Each user is the only authority competent in operations. Also, this information should be avoided by malicious people to reach. Only by malicious people to seize the information may not be objective. They can make the system or application may be inaccessible. Therefore, this requires software security and realibility. Therefore, this study aims web software applications, whereas the threats and vulnerabilities, how to be mentioned steps might be taken.

Keywords: Information security, Information Security Management System (ISMS), Web Application Security, Mobile Application Security.

1. Giriş

Internet, teknik olarak, birçok bilgisayarın ve bilgisayar sistemlerinin birbirine bağlı olduğu, dünya çapında yaygın olan ve sürekli büyüyen bir iletişim ağıdır [1,2].

Internetin icatı ve kullanılmaya başlanması ile bilgisayarlar için kötü niyetli yazılımlar geliştirilmeye başlandı ve bilgisayarlarda yayılmaya başladı. The Creeper virüsü ilk olarak ARPANET'te 1970'lı yılların başında tespit edilmiştir.Tenex işletim sistemi yoluyla yayılmış ve bilgisayara bağlı modemi diğer bilgisayarlara bağlanmak ve onları enfekte etmek için kullanmıştır [3].

Morris Solucanı veya **Internet Solucanı**, Internet yoluyla yayılan ilk solucanlardan birisidir. Üniversitesinde öğrenci olan, Robert Tappan Morris tarafından yazılmış ve MIT'ten 2 Kasım, 1988 tarihinde Internet'e bırakılmıştır. Virüsün MIT'ten bırakılmasının sebebi solucanın Cornell'den geldiğinin anlaşılmaması içindir. Solucanın yazarına göre Morris solucanı zarar vermek amaçlı değil, Internet'in büyüklüğünü tahmin etmek amaçlıydı [3]. Olaydan sonra, Internet üzerinde bilgisayar güvenlik olaylarına tepkinin nasıl iyileştirilebileceği konusunda bir toplantı düzenlendi.

Toplantı sonucunda Internet güvenlik problemleri ile ilgili tek bir iletişim noktası olması gerektiğine ve bu noktanın güvenlik bilgileri için güvenilir bir kaynak olması önerildi. Öneriye karşılık CERT Coordination Center, Internet'te meydana gelen güvenlik olaylarına müdahale etmek amacıyla kuruldu [3].

İnternetin yaygınlaşması ile bilgi paylaşımı artmıştır. Dolayısıyla bilgi önceki zamanlarda söylendiği gibi güç olmaktan çıkmış günümüzde bir varlık halini almıştır. Bilgi ve destek süreçleri, sistemler ve bilgisayar ağları artık önemli ticari varlıklardır. Günümüzde bilginin gizliliği(sadece kullanıcıya izin verilen bilgiye erişim sağlama), bütünlüğü(verilerin yetkisi olmayan kullanıcılar tarafından tahrip edilmemesini ve değiştirilmemesini sağlamak) ve erişebilirliğini (ihtiyaç duyulması halinde yetkisi olan kullanıcıların sistem ve verileri kullanabilmesi) sağlamak çok önemlidir. Bilgi güvenliği bu unsurlara dayanmaktadır.

Bilgi güvenliği; iş devamlılığı, kaçınılmaz felaket durumlarında kaybın en aza indirilmesi, firmaların yapı taşları sayılan kaynakların her koşulda gizliliğinin, ulaşılabilirliğinin ve bütünlüğünün korunması amaçlarını taşır.

Bu nedenle çalışmanın devamında bilginin güvenliği için tehditlerden, güvenlik boşluklarından ve buna karşın alınabilinecek yöntemlerden bahsedilecektir.

Tehditler

Tehdit, bir sistemin veya kurumun zarar görmesine neden olan ve istenmeyen bir olayın arkasındaki gizli neden, olarak tanımlanabilir. Her tehdidin bir kaynağı ve bu kaynağın da yararlandığı sistemdeki bir "güvenlik boşluğu" vardır. Tehditler, tehdit kaynağı açısından bakıldığında iki gruba ayrılarak incelenebilir [4]:

1. İnsan Kaynaklı Tehditler: Bu tür tehditleri de kendi içinde iki alt gruba ayırabiliriz [4]:

a. Kötü niyet olmayan davranışlar sonucu oluşanlar

b. Kötü niyetli davranışlar sonucu

2. Doğa Kaynaklı Tehditler: Bu tür tehditler genellikle önceden tespit edilemezler ve büyük bir olasılıkla olmaları engellenemez. Deprem, yangın, su baskını, sel, ani sıcaklık değişimleri, toprak kayması, çığ düşmesi bu tür tehditlere örnek olarak verilebilir.

Güvenlik Boşluğu (Vulnerability)

Güvenlik boşluğu (Vulnerability), sistem üzerindeki yazılım ve donanımdan kaynaklanan ya da sistemi işletim kuralları ve/veya yönergelerindeki açık noktalar ve zayıf kalmış yönlerdir. Bir güvenlik boşluğu sayesinde bir saldırgan, sistemdeki

bilgisayarlara ya da bilgisayar ağı üzerindeki kaynaklara yetkisiz erişebilir. Bir sunucu bilgisayar üzerinde çalışan bir hizmet (örneğin web sunucu ya da e-posta alma/gönderme hizmeti), modem üzerinden içeri doğru sınırlandırılmamış arama hizmeti, bir güvenlik duvarı üzerinde açık unutulmuş bir erişim noktası (port) güvenlik boşluklarına örnek olarak verilebilirler [4].

Güvenlik boşlukları her kurum için önemli bir güvenlik zaafiyetidir. Bu nedenle her kurum kendi güvenlik açıklarını kapatmak için yatırım yapmaktadır. Dünyada uygulamalara yapılan saldırılar ve yatırım yapılan alanlar maliyeti Gartner'a göre zaafiyetlerin %75'i web uygulamalarında fakat güvenlik için harcanan paranın %90 ağ güvenliği üzerine yapılmaktadır.

Yazılım ya da donanımdan kaynaklanan güvenlik boşlukları, program üreticisi ya da başka bir kaynak tarafından geliştirilen bir "yama program" yardımıyla kapatılmalı ve eldeki yazılım ve donanımların üreticilerinin yayınladığı yama listeleri sürekli olarak takip edilmelidir ve çıkan yamalar vakit geçirilmeden sisteme uygulanmalıdır. Tehditler, bilgisayar sistemlerindeki güvenlik boşluklarına yönelik olarak tanımlanırlar. Yani bir güvenlik boşluğu ortadan kaldırılırsa ya da "yama program" yardımıyla düzeltilirse, söz konusu tehdit ortadan kaldırılır. Bir web uygulamasının güvenliğini sağlamak için uygulamada yer alan süreçleri güvenliğinin sağlanması gerekmektedir [4].

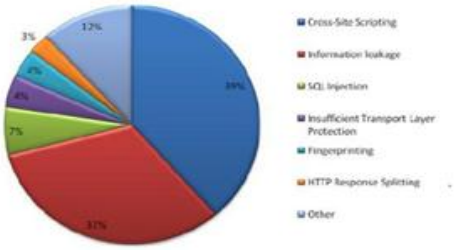
Güvenli bir bilgi sistem ortamı oluşturmak için, ağ ve sistemlerin ve yazılımların düzenli olarak test edilmesi, mevcut zafiyetlerin belirlenmesi ve giderilmesi bir zorunluluktur. Ağ ve sistem güvenliğini sağlamak için bir çok ürün bulunmaktadır. Bu konuda yetişmiş nitelikli elemanlar bulunmaktadır. Zafiyet testleri yapabilecek araçların olması, güvenlik ve ağ yöneticileri için büyük kolaylıklar sağlamaktadır. Bu tür araçlar sayesinde herhangi bir sorun yaşanmadan açıkların giderilmesi mümkün olabilmektedir. Genellikle bir ağdaki güvenlik açıklarını ortaya çıkarmak için ilk işlem olarak ağ ve sistem özelliklerini ortaya çıkaracak bir zafiyet testi yapılır. Bu test, işletim sistemleri, dosya sistemleri, paylaşımlar, kullanıcı hesapları, kullanıcı hakları, açık olan portlar, yazılım güncellemeleri, güvenlik yamaları, anti-virus programları, güvenlik duvarı gibi birçok verinin kontrol ve analiz edilmesine dayanmaktadır.

2. Web Yazılımı Güvenliği

Sistemler başlangıçta güvenlik açısından ne kadar dikkatli kurulursa kurulsun insanlardan veya bilgisayarlarda çalışan yazılımlardan kaynaklanan çeşitli sebeplerle güvenlik açıklarının oluşması her zaman olasıdır. Bu nedenle yazılım ve sistemlerin güvenliğinin sağlanması için ağların ve yazılımların düzenli olarak kontrol edilmesi çok önemlidir. Yazılımı gerçekleştirilen uygulamaların nasıl istenilen gereksinimlerin sağlanması için testi yapılıyorsa güvenlik testlerinin yapılması gerekmektedir. Bu uygulamanın ve bilginin güvenliğinin sağlanması için gerekli bir süreçtir. Uygulamanın yazılım zafiyetleri açısından güvenli bir yazılım haline getirmek için zafiyet açıkları bilmek gerekmektedir. Bu yazılım zafiyetleri her yıl değişmektedir. Örneğin;2010 yılı için dünyada yer alan ilk 10 Web Uygulama Güvenlik Riskleri aşağıdaki gibidir [6].

A1:Enjeksiyon Açıkları
A2:Siteler Arası Betik Yazma
A3:İhlal Edilmiş Kimlik Doğrulama ve Oturum Yönetimi
A4:Güvensiz Doğrudan Nesne Referansı
A5:Siteler Ötesi İstek Sahteciliği
A6: Yanlış Güvenlik Yapılandırılması
A7: Güvensiz Şifreleme ile Depolama
A8:URL Erişim Kısıtlamasına Uyulmaması
A9:Yetersiz Transport Layer Koruma
A10: Kontrol Edilmemiş Yönlendirme

Bu risklerin kullanılarak uygulamalara saldırma oranları şekil-1 [6] gösterilmektedir.



Şekil-1

A1. Enjeksiyon Açıkları (Injection):
SQL injection; saldırganın programa sql sorgu komutlarını enjekte ederek veritabanından izni olmayan istediği bilgileri almasıdır [7]. Örneğin, kullanıcı adı ve şifresi ele geçirilmesi gibi.

A2. Siteler Arası Betik Yazma (Cross-Site Scripting (XSS)):
XSS en kısa tanımıyla istemcinin tarayıcısında istenilen html/dhtml/css veya javascript gibi kodun izinsiz çalıştırılmasıdır. 3 çeşidi bulunmaktadır [8];

- Reflected XSS
- Stored XSS
- DOM-based XSS

A3. İhlal Edilmiş Kimlik Doğrulama ve Oturum Yönetimi(Broken Authentication and Session Management)

Uygulamayı kullanmak isteyen kullanıcının tanımlı bir kullanıcı olup olmadığının kontrolüdür. Oturum anahtarlarının korunmaması ya da düzgün bir şekilde doğrulanmamasıdır [9].

A4. Güvensiz Doğrudan Nesne Referansı (Insecure Direct Object References):
Programda kullanıcının gönderdiği parametreleri saldırganın kendi istediği şekilde değiştirerek göndermesidir. Örneğin, herhangi bir kontrol mekanızması olmadan kod içerisinden bir dosyanın dahil edilebilmesi, direk veritabanı erişim bilgilerinin kod içerisinde saklanması ve çağrılması gibi hatalardır. Saldırgan bu bilgileri kullanarak önemli, gizli dosyalara ve kaynaklara erişebilir.

A5. Siteler Ötesi İstek Sahteciliği(Cross-Site Request Forgery(CSRF)):
Kullanıcının bilgisi olmadan, sanki kullanıcıdan geliyormuşçasına kullanıcının ziyaret etmekte olduğu siteye isteklerin yollanması şeklinde gerçekleştirilen saldırı şeklidir. Örneğin banka hesabındaki paranın sahibi yerine saldırgana aktarılması gibi [10].

A6: Yanlış Güvenlik Yapılandırılması(Security Misconfiguration):
Saldırganın sisteme yetkisiz erişim veya bilgi elde etmek için varsayılan hesapları, kullanılmayan sayfaları, yamasız kusurları, korumasız dosyaları ve dizinleri kullanarak sisteme sızmasıdır. Örneğin evinizdeki ADSL modemleri şifrelerinin değiştirilmemesi gibi.

A7. Güvensiz Şifreleme ile Depolama (Insecure Crytographic Stroge):
Verilerin şifrelenmeden zayıf sunucularda tutulmasıdır. Eğer veriler güçlü bir şifreleme yöntemi ile şifrelenerek veritabanında tutulursa saldırgan veriyi ele geçirmiş olsa bile şifreyi çözemeyeceği için bilgiye ulaşamamış olacaktır.

A8. URL Erişim Kısıtlamasına Uyulmaması (Failure to Restrict URL Access):
Saldırganın sadece bir sayfanı URL'sini değiştirerek programda yetkisi olmayan kısımlara erişebilmesidir. Diğer bir deyişle anonim kullanıcıların korumalı olmayan özel sayfalara erişebilmesidir.

A9. Yetersiz Transport Layer Koruma (Insufficient Transport Layer Protection):
Uygulamaların bir çoğu şifreleme, yetkilendirme, erişilebilirlik gibi güvenlik seviyelerinde zayıftırlar. Örneğin süresi dolmuş sertifikaları kullanılması, zayıf şifrelemele algoritmaları ile iletişim kurulması [10].

A10. Kontrol Edilmemiş Yönlendirme (Unvalidated Redirects and Forwards):
Bir çok web uygulamasında ilk ve son veri kontrolü yapılmaksızın yönlendirmeler yapılmaktadır. Buna en güzel örnek yıllar önce alışveriş sitelerinde ödeme sayfasına geçmeden bir önceki sayfalarda fiyatlar ve miktarlar değiştirilerek ödeme sayfasına yönlendirilmesidir [10,11].

3. Mobil Yazılımı Güvenliği

Bugün cep telefonu kullanımı oranı oldukça fazladır. Özellikle akıllı cep telefonlarının kullanılmaya başlanması ile mobil uygulamalar hızla artmaktadır.Bu uygulamalar her yaştan insanlar tarafından kullanılmaktadır. Bu uygulamaların hızla artması önemli bir güvenlik riski yaratmaktadır. Bu nedenle hem bu platformda yazılım geliştirenlerin hem de kullanıcıların bu konuda daha dikkatli olmaları gerekmektedir. Günümüzde birçok akıllı telefon platformaları vardır. Bu platformlar (Symbian, Android, iPhone, Windows Mobile vb.). Mobil uygulaması riskleri 2 ana kategoriden oluşmaktadır.

a.Kötü Amaçlı İşlevsellik
Kullanıcının oyun veya yardımcı bir programı telefonuna yüklediğini sanarak gizli Truva uygulaması telefonuna yüklemesidir. Bu uygulama spyware, phishing veya casus yazılım olabilmektedir. Genellikle bu yazılımlar aşağıdaki amaçları gerçekleştirmek için kullanılıyor;

- Kullanıcıyı etkin bir şekilde izleme ve kullanıcının telefonundan veri alma
- Yetkisiz arama, SMS, ve ödemeler
- Yetkisiz ağ bağlantısı
- UI kimliğe bürünme
- Sistem modifikasyonu yapmak (rootkit, APN proxy config)

b.Güvenlik Açıkları
Tasarımda veya uygulamada var olan açıklara saldırganlar tarafından saldırılarak mobil cihaz verilerini ele geçirme şeklinde olmaktadır. Güvenlik açıklarından yararlanılarak cihazdan yetkisiz erişim ile uygulamaların bilgilerine ulaşılmaktadır. Güvenlik açıkları aşağıda sıralanmıştır.

- Veri sızıntısı (yanlışlıkla ya da yan kanal)
- Güvensiz önemli verileri depolama
- Güvensiz veri iletimi

Mobil yazılım geliştiriciler uygulama geliştirirken kısıtlı bellek, işlemci ve pil gücü gibi sorunların yanında kullanıcıların verilerini güvende tutmak gibi konuları da dikkate almaları gerekmektedir.

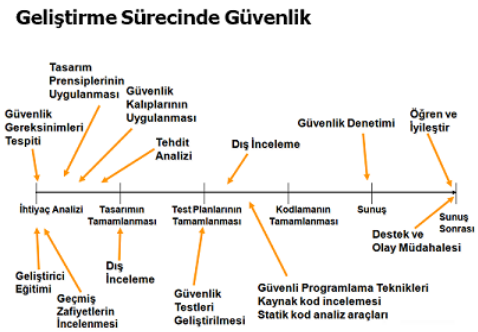
Geliştiriciler, uygulama ile cihazın içine herhangi bir kullanıcının veri girişlerini de korumalıdır. Ayrıca kötü amaçlı yazılım uygulamaların yüklenmesini zorlaştırmak için akıllı kart platformlarda özel izinlere veya ayrıcalıklara erişmek için izin gerekecek şekilde düzenleme yapması gerekmektedir.

4.Güvenli Yazılım Geliştirme Metodları

Güvenlik kavramının bir ürün değil, bir süreç olduğu unutulmamalıdır. Bütün güvenlik ürünlerinin alınması ile sistem güvenliği sağlanmış olmayacağı bilinciyle güvenlik sürecinin yazılım geliştirme sürecine dahil edilmesi gerekmektedir.

Yazılım geliştirme süreçleri olan planlama, analiz, tasarım, kodlama, test ve kurulum gibi adımlar güvenlik sürecide içermelidir. Güvenli Yazılım Geliştirme Süreçleri (secure software development life cycle) sayesinde tehdit modelleme, risk analizi, yazılımcılar için güvenlik eğitimleri ve dokümanları, mimari analiz, kod analizi, sızma testleri (*penetration test*) gibi güvenlik için vazgeçilmez gereksinimler bir bütün olarak yazılım sürecine entegre edilmelidir [12].

Şekil-2 [13] görüldüğü gibi bir geliştirme süreci ve bu süreç bağlamında gerçekleştirilebilecek güvenlik aktiviteleri görülmektedir.



Şekil-2

Düşey eksende ortasında soldan sağa doğru

tipik bir geliştirme sürecinde yer alan aktiviteler verilmiştir. İhtiyaç analizi, tasarım, test planlaması, kodlama, sunuş ve sunuş sonrası aktiviteleri bu sırada gösterilmiştir.

Geliştirme sürecini gösteren çizginin altında ve üstünde ise güvenlik ile ilgili aktiviteler yer almaktadır. İhtiyaç analizi aşamasında güvenlik gereksinimlerinin tespit edilmesi, geliştirme ekibinin güvenlik konusunda temel eğitim almış olmasının sağlanması, benzer uygulama alanlarında farklı ekiplerin geliştirdikleri yazılımlardaki geçmiş zafiyetlerinin incelenmesi, güvenli tasarım prensiplerinin ve tasarım güvenlik kalıplarının uygulanması ve yazılımın tehdit modellemesinden geçirilmesi gibi çeşitli teknikler ihtiyaç analizi ve tasarım aşamasında uygulanabilmektedir [13].

Geliştirme aşamasından hemen önce tasarımın bağımsız bir kuruluş tarafından bağımsız incelemeye tabi tutulması, testler kapsamında güvenlik testlerinin de planlanması ve uygulanması, güvenli programlama tekniklerinin uygulanması, kaynak kodların kontrol altında tutulması ve analiz araçları ile güvenlik zafiyetlerine karşı sınanması ve penetrasyon testleri sunuş öncesinde uygulanabilecek tekniklerdir [14, 15]. Yazılım projelerinde bu aktiviteler değerlendirmeli ve geliştirme sürecine entegre edilmesi mümkün olanlarının sürece dahil edilmesi sağlanmalıdır.

Web ve mobil yazılım güvenliğini sağlamak için hem yazılımı geliştirenin hem de uygulamanın sistem tasarımını yapan kişilerin kendilerini saldırganın yerine koymalı ve oluşabilecek açıklıkları belirlemeye çalışmalıdır. Uygulamanın istenilen gereksinimler için test edilmesinin yanı sıra kullanıcının istenmeyen durumlarda uygulama ile yapılabilirliğine bakmak için testinin iyi yapılması gerekmektedir. Her türlü düşünülmeyen durumlar için testlerin yapılması gerekmektedir. Ayrıca test yapılırken kod analizi yapılmalıdır. Tekrarlanan gereksiz kod yazılmamaya çalışılmalıdır.

Yazılım geliştirme dillerinin erişim kısıtlama özellikleri kullanılmalıdır. *Public, protected, private, static, final* gibi kavramlarının yazılım geliştirilirken sınıf, metod, foksiyon, değişkenler için kullanılması gerekmektedir. Değiştirilmesi istenmeyen değişkenler sabit (*constant*) olarak tanımlanarak değiştirilmesi engellenebilmektedir. Alt sınıfı olmayan sınıfları ve başka bir alt sınıfta tekrar tanımlanmayacak olan metotların *final* olarak tanımlanmasına dikkat edilmelidir [16].

Sınıfların değiştirilemez olarak tasarlanmasına gayret edilmelidir. Bu sayede yapılan her değişiklik bu değişkenin yeni bir kopyasının oluşturulmasıyla sonuçlanır. Şayet değiştirilemez tasarlamak mümkün değilse, sınıfın değişkenliğini mümkün olduğunca kısıtlanması gerekmektedir. Ayrıca değiştirilebilen (*mutable*) sınıfınızın güvenli bir şekilde kopyasını almayı sağlayan bir metodu da kullanılmalıdır [16].

Erişim kontrollerinin iyi yapılandırılması gerekmektedir. Uygulamaların bir dosyayı okuma/yazma hakkı, sistemin hangi özellik (*property*) değişkenlerini okuma/yazma hakkı, programı sonlandırma hakkı ve ağ üzerinden hangi bilgisayara erişim hakkı olması gerektiğinin kontrollerinin iyi yapılması gerekmektedir. Erişim hakları verilirken en düşük erişim hakkı uygulanmalıdır. İhtiyaç olmayan erişim hakkı verilmemelidir [16]. Aynı şekilde uygulamaların hangi veritabanına ve hangi diğer uygulamalara erişmesi gerektiği iyi ayarlanmalıdır. Sadece yazılımı gerçekleştirilen uygulamanın güvenli olması yeterli değildir. Uygulamanın bağlantısı olan diğer sistemlerle olan bağlantılarınında güvenli olması gerekmektedir [17].

Yazılımı yapılan uygulamalarda kullanıcının ekrandan girdiği alanlar için girdi kontrollerinin yazılımcı tarafından yapılması

gerekmektedir. Kullanıcının girdiği değerleri kontrol etmesi gerekmektedir. Tanımladığı kuralların dışında kalan girdileri kabul etmemesi ve kullanıcıyı yönlendirecek şekilde bilgi mesajı vermelidir. Ayrıca web ve mobil uygulamalarda webservis kullanılmaktadır. Dolayısıyla kullanılan webservislerinde güvenlik kontrollerinin yapılması gerekmektedir. Çünkü webservisten kaynaklı güvenlik açığı uygulamanın da güvenliğini riske atacaktır. Webservislerin güvenliğinin sağlanması için standartlar bulunmaktadır [18].

Yazılım güvenli olması için uygulamalarda mutlaka olması gerekenler kısımlar :

- Kimlik Doğrulama
- Yetkilendirme
- Oturum Yönetimi

Kimlik Doğrulama:
Uygulamayı kullanacak kullanıcıları tanımlamak ve kendi verilerine erişim izinlerini doğrulamak için kullanılan işlemdir. Kimlik doğrulama bir uygulama için en önemli yapıdır. Uygulamayı kimin kullanıp kimin kullanmayacağına bu yapı ile karar verilir. Bu nedenle yazılımcı kendi uygulaması için en uygun kimlik doğrulama sistemini seçmelidir. Kimlik doğrulamanın doğru çalışması için güçlü bir şifre üretim yapısı olmalıdır. Güçlü bir şifre üretimi için de bir şifre politikası olması gerekmektedir. Kullanıcı şifre politikasına uygun şifre oluşturması için mecbur bırakılmalıdır. 8-16 karakter arası parolaların kırılması çok kolay değildir.

Yetkilendirme:
Kullanıcının izin verilen işlemlerin yetki derecelerine uygun olarak kullanıcıların sadece uygulamada yetkili oldukları işlemleri gerçekleştirilebilmesini sağlamaktır. Kullanıcı hesapları uygulama içinde kendilerine atanan görevleri yapacak şekilde en düşük haklara sahip olmalıdırlar.Kullanıcılar yetkilendirilmiş veya yönetimsel hiç bir fonksiyonu kullanamamalıdırlar [19].

Kullanıcıları sistem seviyeside yetkilere sahip olurlarsa(Admin,root vb) gibi çekirdek seviyesinde çalışan uygulamaları yüklemeleri sağlanabilir. Bu uygulamalar Rootkitlerdir. Bu nedenle bu yetkilerin kullanıcılara verilmemesi gerekmektedir.

Oturum Yönetimi:
Kimliği doğrulanmış kullanıcıların açtıkları oturumları ile sağlam ve şifreli olarak güvenli bir bağlantı olmasıdır. Açılan oturumu sadece kullanıcının kullanmasını sağlayacak şekilde oluşturulmasıdır. İyi bir oturum kimlik bilgisinin iki kritik özellikleri vardır. Bu özellikler rasgelelik ve uzunluğudur. Oturum yönetiminde oturum kimliğinin içeriğinin beklenen boyutta ve türde olduğunun kontrollünün oturumun doğrulama işlemini yapmadan önce yapılması gerekmektedir. Çünkü çok geniş oturum kimliği bilgisi arabellek taşması tipi saldırı teşkil edebilir. Ayrıca, oturum kimlik içeriği beklenmedik bilgi içermediğinden emin olunmalıdır [19].

5.Sonuç
Yazılım güvenliği ister web tabanlı uygulamalar için olsun ister mobil tabanlı uygulamalar için olsun sürekli değişen ve gelişen bir süreçtir. Çünkü devamlı olarak saldırganlar uygulamalardaki açıkları bulmak için çalışmaktadır. Bugün bilinen açıklara ve tehditlere karşı savunmalarını sağlayacak birçok yöntem bulunmaktadır. Fakat hiçbir savunma yöntemi uygulamaları %100 güvenli hale getirmemektedir. Bu nedenle uygulamaların ve sistemlerin devamlı kontrol halinde tutulması gerekmektedir. Düzenli olarak testlerin yapılması gereklidir. Değişen ve gelişen saldırı tiplerine ve risklerine karşı uygulamaları korunaklı hale getirmelidir.

Ayrıca devamlı olarak yeni saldırı tipleri ortaya çıkmaktadır. Güvenli bir uygulama geliştirmek için devamlı araştırma yapılmalıdır. Ortaya çıkan açıklık ve zafiyetler için yazılımda savunma mekanizmaları oluşturulmalıdır.

Sonuç olarak yazılım da güvenliğin sağlanmasının ilk aşaması insanlarda bu konuda farkındalık oluşturulması ve bu alanda kendilerini geliştirmeleri için yönlendirilmelidir. Çünkü ancak bu kişiler güvenli yazılımlar geliştirilebilir.

6.Kaynaklar

[1] Internet : **htpp:://yunus.hacettepe.edu.tr/~sadi/dersler/ebb/eb467-guz200/umut-p.html,** (E.T: 01.12.2012).

[2] Internet :**http://www.internetarsivi.metu.edu.tr/tarihce.php,**(E.T: 01.12.2012).

[3] Mert ÜNERİ, Bilgi Teknolojileri Güvenliği Geçmişi – Geleceği, **TÜBİTAK-UEKAE.**

[4] Bilişim Güvenliği**, Pro-G Proje Bilişim Güvenliği ve Araştırma San. ve Tic. Ltd. Şti**, (Sürüm 1.1).

[5] Internet: **http://www.gartner.com/Technology,**(E.T: 01.12.2012).

[6] Internet: The Ten Most Critical Web application Security Risk, OWASP Top10-2010, **https://www.owasp.org/,** (E.T: 01.12.2012).

[7] V. Benjamin Livshits and Monica S. Lam, Finding Security Vulnerabilities in Java Applications with Static Analysis, **Computer Science Department Stanford University.**

[8] G. JOURDAN, Software Security Vulnerabilities Seen As Feature Interactions, **School of Information Technology and Engineering, University of Ottawa.**

[9] E. Karaarslan, M. Sürücü, N. Karadağ, O. Yalçın, V. Fetah**, OWASP İlk 10, En Kritik 10 Web Uygulaması Güvenlik Zayıflıkları**, 2007.

[10] Internet:M. Kaya, En Güncel 10 Güvenlik Riski, **http://www.olympos.net/belgeler/owasp/en-guncel-10-guvenlik-riski-5071048.html,** (E.T: 01.12.2012).

[11] D. Gollmann, Securing Web applications, **Information Security Technical Report** (2008)1-9, 2008 Elsevier Ltd.

[12] A. AlAzzazi, A El Sheikh, Security Software Engineering: Do it the right way, **Proceedings of the 6th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems, Corfu Island, Greece**, February 16-19, 2007.

[13] B. Dayıoğlu, Yazılım Geliştirme Yaşam Döngüsü ve Güvenlik, **Pro-G Bilişim Güvenliği ve Araştırma Ltd. Şt**i.,2008.

[14] Y. Huang, C. Tsai, T. Lin, S. Huang , D.T. Lee, Sy-Yen Kuo, A testing framework for Web application security assessment, **Computer Networks** 48 (2005) 739–761.

[15] N. Li, T. Xie, M. Jin, C. Liu, Perturbation-based user-input-validation testing of web applications, **The Journal of Systems and Software** 83 (2010) 2263–2274.

[16] Internet: E. İslam Tatlı, Java'da Güvenli Yazılım Geliştirme, http://www.arhitectingsecurity.com(E.T: 01.12.2012).

[17] Java Access Control Mechanisms, **http://labs.oracle.com/techrep/2002/smli_tr-2002-108.pdf,** (E.T: 01.12.2012).

[18] C. Geuer-Pollmann, J. Claessens, Web services and web service security standards, **Information Security Technical Report** (2005) 10, 15-24.

[19] B. URGUN, Web Uygulama Güvenliği Kılavuzu, **Ulusal Elektronik ve Kriptoloji Araştırma Enstitüsü**, SÜRÜM 1.00, 16 KASIM 2007