GÜVENLİ YAZILIM GELİŞTİRME TEMEL KURALLARI DOKÜMANI

SÜRÜM 1.12

03 Şubat 2014

Hazırlayan

TÜBİTAK BİLGEM Siber Güvenlik Enstitüsü

P.K. 74, Gebze, 41470 Kocaeli, TÜRKİYE
Tel: (0262) 648 1000
Faks: (0262) 648 1100
http://sge.bilgem.tubitak.gov.tr
http://www.bilgiguvenligi.gov.tr
sge@tubitak.gov.tr

Bu Sayfa Boş Bırakılmıştır.

AMAÇ ve KAPSAM

Ulusal Siber Güvenlik Eylem Planı 10 numaralı maddesinin alt eylemlerinde biri olan güvenli yazılım geliştirme temel kurallarının oluşturulması kapsamında, kritik altyapılar için geliştirilen yazılım projelerinde yazılım geliştirici ve proje yönetim ofisi tarafından rehber olarak kullanılmak üzere temel güvenlik kurallarının hazırlanması istenmiştir.

Bu doküman yazılım geliştirme projelerinin her bir fazında etkili olarak kullanılabilir. Öncelikle yazılım projelerinin ilk fazında yazılım geliştiricilerin eğitimi kapsamında bu dokümandaki kuralların üzerinden geçilmesi ve her bir kuralın yazılım geliştiriciler tarafından anlaşıldığından emin olunması faydalı olacaktır. Yazılım projesinin erken evrelerinde, geliştiricilerin ve proje yönetim ofisinin proje boyunca izlenecek metodu, dokümanda belirtilen kuralların nasıl ve ne zaman uygulayacaklarını belirlemeleri, proje sonunda kullanıma sunulacak uygulamanın güvenliği açısından önemlidir. Ayrıca projenin tasarım fazının tamamlanması sonrasında güvenlik tasarım dokümanını oluşturulurken de bu temel kurallar dokümanına başvurulması faydalıdır. Bu şekilde tasarlanan sistemde ya da uygulamada, uygun kılavuz maddelerinin diğer proje görevleri ile birlikte sorumlulara atanarak izlenmesi başarıyı artıracaktır. Proje yönetim ofisinin izlenmesinde fayda gördüğü kılavuz maddelerinin, teknik ekip tarafından uygulanmaları sistemin güvenli hale gelmesini sağlayacaktır.

Bu dokümanda verilmiş her bir kural detaylandırılmış ve gerektiğinde kural için açıklama yapılmıştır. Fakat konunun tam anlaşılır olması için yukarıda da belirtildiği üzere, eğitim kapsamında bu kuralların tek tek ele alınması, gerekiyorsa verilen referansların takip edilerek konunun detayına hâkim olunması yoluna gidilmelidir.

Ulusal Siber Güvenlik Eylem Planı 10 numaralı maddesinin diğer bendinde ise kritik altyapılar için geliştirilen yazılımların güvenlik değerlendirmeleri kapsamında ilgili kurumların bünyesinde gerekli teknik isterlerin uygulanması ve kontrolü için kullanılabilecek güvenlik isterleri havuzu oluşturulması öngörülmektedir. Güvenlik isterleri daha çok yazılımı talep eden kritik alt yapı sahibi kurum ve kuruluşun uygulamadan beklediği güvenlik özelliklerini belirlerken temel güvenlik kurallarını içeren bu doküman, yazılımı geliştiren proje ekibine yönelik olarak düşünülmelidir.

Doküman Yapısı

Bazı özel terimler ve kısaltmalar, ilk kullanıldıkları yerde "" ile işaretlenmiş ve italik yazılmışlardır. Bunlar dokümanın sonunda SÖZLÜK bölümü altında ele alınmış ve açıklamaları yapılmıştır.*

Güvenli Yazılım Geliştirme Temel Kurallar Dokümanı, bir uygulama için, programlama dilinden bağımsız olmak üzere, dokümanın oluşturulmasından, yazılımın tasarlanmasına, kodlanmasına, test edilmesine, kurulumuna ve kabul muayenesine kadar geçen süreçte, bu sürece müdahil tüm paydaşların yararlanabilecekleri, konulara bölünmüş kurallar dizininden oluşmaktadır. Yazılımı güvenli bir şekilde geliştirme adına paydaşlar tarafından hedeflenmesi gereken, bu kuralların tümünün yazılımda yer bulmasından ziyade yazılımın ve yazılımın kurulacağı ortamın ihtiyaçlarına göre bu dokümandaki kuralların incelenmeleri, sonrasında gerçeklenmeleri veya gerçeklenmelerine zemin hazırlamasıdır.

Dokümandaki konu başlıkları, yazılım güvenliğinde önemli ve dikkat edilmesi gereken temel hususlar etrafında oluşturulmuştur. Bu başlıklar ve kısa özetleri şu şekildedir;

- **Verinin Korunması**

  Bu başlık, verinin yetkisiz kişilerin eline geçmesini engellemeyi, verinin hem bilgisayar sistemlerinde, hem saklama ortamlarında, hem de ağ üzerinde gönderici ve alıcı arasında taşınırken yetkisiz erişimlerden korunması, gizliliği* ve bütünlüğü* ile alakalı hususları içermektedir.
- **Kimlik Doğrulama***

  Bu başlıkta, kullanıcının sisteme/uygulamaya bağlanabilmesi veya uygulama içindeki birimlerin birbirlerini tanımaları için yapılması gereken işlem olan kimlik doğrulamanın güvenlik kuralları işlenmektedir.
- **Yetkilendirme***

  Bu başlıkta kimlik doğrulaması sağlanan kullanıcıların/yazılım birimlerinin sisteme, programa ve ağa hangi yetkilerle erişim hakkına sahip olduklarının belirlendiği yetkilendirme mekanizmaları ve bu mekanizmaların güvenlik unsurları işlenmektedir.
- **Erişilebilirlik***

  Bu başlıkta verinin her an ulaşılabilir ve kullanılabilir olmasını amaçlayan erişilebilirlik prensibi ele alınmaktadır. Bu kapsamda bilişim sistemlerinin kendilerinden beklenen iş sürekliliği ve bunun bilgi güvenliğine olan yansımaları madde madde işlenmektedir.

- **Sistem İzleme ve Denetimi***

  Bu başlıkta, yazılımda her hangi bir sorun veya anomali ile karşılaşıldığında sorunun tespiti için kullanılan sistemler, bunların yazılım güvenliğindeki yerleri işlenmektedir.

- **Diğer Güvenlik Önlemleri**

  Bu başlıkta yazılımlara en sık gerçekleştirilen saldırılara karşı alınacak önlemler irdelenmektedir. Bu önlemlerin çoğu küçük kod örnekleri ile açıklanmışlardır.

Bu konu başlıkları içindeki maddeler diğer konu başlıkları ile paralellik gösterebilir. Bu durumu göz önünde bulundurarak, dokümanın okunabilirliğini artırmak için her maddenin altına etiketler yerleştirilmiştir. Bu etiketlerden yola çıkarak, o madde ile alakalı tüm güvenlik başlıklarını görülebilir.

1. VERİNİN KORUNMASI

1.1 Uygulama, ister veya tasarım dokümanlarında aksi belirtilmedikçe her türlü verinin gizliliğini korumalıdır.

Etiket: Gizlilik

Neden: Siber güvenlik kapsamında değerlendirilecek olan veri sadece uygulamanın son kullanıcılara sunduğu veriler değildir. Veri tanımını, saklanan veri ve taşınan veri olmak üzere ikiye ayırmak mümkündür. Taşınan verinin gizliliği tasarıma göre değişiklik gösterebilir ve farklı katmanlarda gerçekleştirilebilir.

Nasıl: Verinin taşındığı kanalın gizliliğinin sağlanması için SSL* ya da IPSEC* kullanımını örnek olarak verilebilir. Saklanan veri ise uygulama yazılımı yapılandırma ayarları, uygulama sunucu yapılandırma dosyaları, uygulamalar ve sistemler arası iletişim için kullanılan ve dosya içinde ya da veri tabanında saklanan parolalar ve işletim sistemi üzerinde bulunan ve yazılım tarafından kullanılan parolalar bu kapsamda değerlendirilmelidir.

1.2 Uygulama servisleri "noktadan noktaya" ya da "uçtan uca" gizliliği desteklemelidir.

Etiket: Gizlilik

Neden: "Noktadan Noktaya" (point-to-point) kapsamında, son kullanıcı ile uygulama sunucusu arasında paylaşılan verinin gerekirse başka ara düğümlere (sunuculara) de uğrayarak hedefe ulaşabileceği öngörülmektedir.

Nasıl: Örneğin SOAP* protokolü kullanan bir web servisi SSL kullanmamasına rağmen veriyi şifreleyip SOAP paketi içinde HTTP* olarak gönderebilir. "Uçtan uca" (end-to-end) şifreleme de son kullanıcı ile uygulama sunucu arasında şifreli bir tünelin oluşturulmasını ve trafiğin arada bulunan herhangi bir düğüm tarafından görülemeyeceği anlamına gelmektedir.

Diğer taraftan HTTP dışında başka bir protokol ile iletişim sağlayan uygulamalar tasarım uyarınca noktadan-noktaya ya da uçtan-uca şifreleme desteği sunmalıdır.

1.3 Uygulama kullanıcının kimlik bilgilerini taşıma esnasında korumalıdır. SSL veya benzeri teknolojileri tüm kimlik denetimi süreci esnasında kullanmalıdır.

Etiket: Gizlilik, Kimlik Doğrulama

Neden: Kimlik bilgileri, gizliliği korunması gereken en hassas uygulama verisidir.

Nasıl: SSL protokolünün güncellenmiş son sürümü kullanılmalıdır.

1.4 Uygulama "noktadan noktaya" veya "Uçtan uca" veri bütünlüğünü desteklemelidir.

Etiket: Gizlilik, Veri Bütünlüğü

Neden: Bu kapsamda "noktadan noktaya" (point-to-point) son kullanıcı ile uygulama sunucusu arasında paylaşılan verinin gerekirse başka ara düğümlere (sunuculara) de uğrayarak hedefe ulaşabileceği öngörülmektedir. "Uçtan uca" (end-to-end) bütünlük de amaç, veriyi olması gerektiği şekilde tutmak ve korumaktır. Bilginin bozulmasını, değiştirilmesini, yeni veriler eklenmesini, bir kısmının veya tamamının silinmesini engellemeyi hedefler. Bu durumda veri, haberleşme sırasında izlediği yollarda değiştirilmemiş, araya yeni veriler eklenmemiş, belli bir kısmı ya da tamamı tekrar edilmemiş ve sırası değiştirilmemiş şekilde alıcısına ulaşır.

Nasıl: Örneğin dijital imza*; bir bilginin üçüncü tarafların erişimine kapalı bir ortamda, bütünlüğü bozulmadan (bilgiyi ileten tarafın oluşturduğu orijinal haliyle) ve tarafların kimlikleri doğrulanarak iletildiğini elektronik veya benzeri araçlarla garanti eden bir sistemdir. Bu sistem içinde veri bir uçtan diğer uca giderken, ilk önce SHA*, RIPEMD* vs. gibi özetleme(hash)* algoritmalarının son sürümleri kullanılarak özetlenir. Özetlenen veri, güvenilirliği doğrulanmış bir sertifika otoritesi(örn. KAMUSM) tarafından temin edilen, çift taraflı(asimetrik) bir algoritma(RSA, DSA, ECDSA vs. gibi) ile oluşturulmuş özel anahtar(private key) ile şifrelenir. Bu şifrelenmiş veri özeti veri ile beraber diğer uca gönderilir. Diğer uçta, gönderilen verinin özeti aynı algoritma ile alınır, karşı taraftan gelen imzalı veri özeti, şifrelendiği özel anahtara karşılık gelen açık anahtar ile çözülerek, bu iki özet karşılaştırılır ve verinin bütünlüğü garanti altına alınmış olur. Veri imzalama için gereken ETSI 101733 standardına uyumlu araçlar, sertifikalar KamuSM'den elde edilebilir.

Referans:

- http://www.kamusm.gov.tr/depo/sertifikalar/depo.jsp

1.5 Uygulama içindeki veriler ve çıktılar güvenlik sınıflandırmasına tabii tutulmalıdır.

Etiket: Gizlilik, Yetkilendirme

Neden: Uygulama içindeki veri ve çıktıların farklı gizlilik ve hassasiyet dereceleri olabilir. Özellikle yetkilendirme esnasında farklı gizlilik seviyelerinin gruplandırması önem arz etmektedir.

Nasıl: Örneğin "Tasnif Dışı", "Hizmete Özel", "Ticari Özel", "Gizli" vs. gibi sınıflandırmalar kullanılabilir. Güvenlik tasarım ya da iş tanım dokümanında hangi varlıkların hangi güvenlik sınıflandırmasına tabi tutulacağı belirtilebilir.

1.6 Uygulama başka kaynaklara(örneğin uygulama veri tabanına bağlanırken) erişim için kullandığı parolaları şifrelenmiş(encrypted) bir halde saklamalıdır.

Etiket: Gizlilik, Şifreleme

Neden: Saldırganların ilk ulaşmaya çalışacakları sistem varlığı parolalardır. Dolayısıyla parolalar uygulama içinde korunması gereken en önemli verilerdir.

Nasıl: Parolalar hiçbir durumda uygulamanın kaynak koduna girilmiş olmamalıdır. Parolaların şifrelerini çözmek için gereken anahtarlar, yetkisiz erişimden kuvvetli bir şekilde korunmalıdırlar.

1.7 Uygulama, son kullanıcıların ya da istemci durumundaki uygulama servislerini kullanan diğer sistemlerin kimliklerini doğrulamak için kullandığı parolaları kriptografik özet halinde (hash) saklamalıdır.

Etiket: Gizlilik

Neden: Parolaları ele geçiren bir saldırganın bu bilgiyi kullanamamasını sağlamak için düz metine kıyasla güvenli olduğu bilinen kriptografik özet (Hash) yöntemleri kullanılır.

Nasıl: Örneğin uygulama veri tabanında saklanan son kullanıcı parolaları özet olarak tutulmalıdır. Bu işlemi yaparken güvenliğinde (şimdilik) şüphe olmayan sha3 gibi kuvvetli özet algoritmaları kullanılmalıdır. Parolaların özetleri alınırken, özet fonksiyonuna rastgele bir sayı katılmalıdır(tuzlama). Bu şekilde özeti alınmış parolalar sözlük* ve gökkuşağı* saldırılarına karşı korunmuş olacaklardır.

Referans:

- http://www.bilgiguvenligi.gov.tr/web-guvenligi/parola-analizi.html

1.8 Silinmiş verilere uygulama bileşenleri üzerinden tekrar ulaşım engellenmelidir. Bellekte ye da disk sisteminde oluşturulan nesnelerin (objects)* gizli veri içermesi engellenmelidir.

Etiket: Gizlilik-Veri Bütünlüğü-Veri silme

Neden: Saldırganlar, sadece mantıksal* olarak silinmiş ancak geri döndürülebilecek verileri geri getirerek bunlardan istifade etmeye çalışacaklardır.

Nasıl: Bu önlemle mantıksal olarak silinmiş verilerin fiziksel olarak da silinmesi ve uygulama tarafından erişilemez olduğunun garantilenmesi hedeflenmektedir.

Bellekte tutulan ve uygulamanın normal ya da istem dışı kapatılması sonrasında bellekten silinen veriler bu önlemin kapsamı haricinde tutulabilir. Aynı şekilde iş tanımında aksi belirtilmemişse yedeklenen veriler de bu önemin kapsamı dışındadır.

Referans:

- https://www.bilgiguvenligi.gov.tr/yedekleme/bilgisayar-sistemlerinde-guvenli-dosya-saklama-ve-silme-2.html

1.9 Tasarım uyarınca, gerektiği durumlarda veriler şifrelenirken GİD'de belirtilen standartlara uygun ya da belirtilen otorite tarafından onaylanmış bir kripto ürünü* veya mekanizması kullanılmalı ve bu şekilde depolanmalıdır.

Etiket: Gizlilik-Şifreleme

Neden: Bir standarda bağlı olmayan, güvenliği garanti edilmemiş, zayıf kripto ürünleri, mekanizmaları veya algoritmaları ile yapılan şifreleme işlemlerine güvenilemez. Saldırganların bu tarz, zayıf mekanizmaları hedef alacakları ön görülmelidir. Yazılımın sahibi olduğu kurumun kullanması gereken kripto ürün, algoritma ya da yöntemlerini belirleyen standartlar olabilir. Bu tür bir bağlayıcı standart yoksa ülkede bu konuda çalışan kurumlara ya da yapılmış yayımlara başvurulabilir.

Nasıl: Örneğin, C++ için açık kaynak kodlu Crypto++® Library 5.6.2, SSL uygulamaları için openssl kütüphanesi kullanılabilir.

Referans:

- http://www.openssl.org/docs/crypto/crypto.html

- http://www.cryptopp.com/

1.10 Uygulama içindeki veri akışı kontrol politikası belirlenmeli ve uygulanmalıdır.

Etiket: Gizlilik, güvenli mimari

Neden: Özellikle uygulamanın etkileşim halinde olduğu sistemin sınır güvenliğini sağlamak için önem arz eden veri akışı kontrol politikası, verinin sistem içinde veya sistemler arasında nerelerde dolaşabileceğini düzenleyen politikadır. Veriye kimin erişebileceğin düzenlenmesi ile karıştırılmaması gerekir.

Nasıl: Örneğin hassas bilgilerin internete açık ortamlara çıkmasının engellenmesi, dışarıdan gelen fakat içeriden geliyor gibi gözüken veri trafiğinin bloke edilmesi ve iç web ara vekilinden(Proxy)* den çıkmayan web istemlerinin engellenmesi.

Veri akışı kontrol politikasının gerçeklenmesi için farklı çözümler bulunmaktadır; veri, kaynak ve hedef objelerine özel etiketler atanarak gerçeklenebilir veya korunma altına alınmış alanlar(protected domains) belirlenerek sağlanabilir.

Referans:

- http://www.cis.upenn.edu/~stevez/papers/Zda04.pdf

1.11 Uzaktan çalıştırılabilen veya sistemin değişik parçaları arasında transfer edilen taşınabilir kodlar(mobile codes) dikkatli bir şekilde ele alınmalı, envanteri tutulmalıdır.

Etiket: Mobil Kodlar, Bağımsız Yorumlayıcılar, Applet Güvenliği

Neden: Taşınabilir kodlar, tüm web uygulamalarında sıklıkla kullanılan Javascript veya VBScriptler gibi betik kodları, JAVA apletleri, ActiveX kontrolleri veya Flash animasyonları, Shockwave filmleri veya Microsoft ofis makroları olabilir.

Nasıl: Bu şekilde taşınan kodları erişim kontrolüne kapsamında üründen izole bir alanda (sandbox) çalıştırmak gerekmektedir. Ayrıca bu kodların imzalı olması ve bu imzaların kontrol edilmesi gerekmektedir. Özellikle bu kodların ürettikleri girdileri üründe kullanmadan önce girdi denetimine tabi tutmak önemlidir.

Referans:

- http://seit.unsw.adfa.edu.au/staff/sites/lpb/papers/mcode96.html

1.12 GİD'de uygulama içindeki tüm verilerin korunması öngörülüyorsa, veriler uygulama bileşenleri arasında şifreli olarak (gizliliği korunarak) iletilmelidir.

Etiket: Gizlilik, Güvenli Veri Taşıma

Neden: Saldırganlar uygulama bileşenleri arasına girip MITM saldırıları* ile yapılan veri transferini görebilir ve gizli verileri ele geçirebilirler.

Nasıl: Bunu sağlamak için örneğin veri tabanı bağlantısı yaparken JDBC yerine secure JDBC(JDBCS) veya FTP ile veri aktarımı yapılacaksa FTPS kullanılması tercih edilebilir. İlk risk analizi ya da tehdit modelleme aşamasında iç ağda belirlenen tehditler ve ortaya çıkan risklerin değerlendirmesine göre iç ağ trafiğinin de şifrelenmesi gerekebilir.

1.13 Uygulama, herhangi bir fonksiyonu çalışmaya başlamadan önce güvenlik fonksiyonlarının çalışır ve ayakta olduğunu garanti etmelidir.

Etiket: Erişilebilirlik, Güvenli Mimari

Neden: Saldırganlar, hedefli DOS* saldırılarıyla güvenlik için devreye alınmış uygulama fonksiyon ve modüllerini devre dışı bırakabilirler.

Nasıl: SSL sertifikasının güncelliğini, işletim sistemi dosyalarına erişim yetkilerini, kayıt mekanizması bileşenlerinin çalışır durumda olduğu ya da şifreleme için bir servis kullanıyorsa ilgili servisin ayakta olduğu kontrol edilmelidir.

1.14 Uygulama veri dağıtım servisleri(DDS)* kullanılıyorsa, bunlara kesinlikle gizliliği korunmuş bir kanal üzerinden (Örn. SSL kullanarak) erişmelidir.

Etiket: Gizlilik, Güvenli Veri Taşıma

Neden: Dağıtık yapıda çalışan servisler gerçekleme gereği olarak güvenlik zafiyetlerine daha açık durumdadır.

Nasıl: DDS, hava trafiği, scada, smart grid gibi kritik yazılımlarında gerçek zamanlı veri paylaşımı için kullanılan dağıtık web servisleridir. Bu servislerle kullanımında veri aktarırken güvenli bağlantı yapılması gerekmektedir.

1.15 Uygulama varlıkları arasında karşılıklı bütünlüğü (referential integrity) sağlamakla yükümlü olmalıdır.

Etiket: Veri tabanı bütünlüğü

Neden: Verinin korunduğu yer olan ver tabanı, veri bütünlüğünün sağlanması gereken en önemli uygulama birimidir.

Nasıl: Karşılıklı bütünlük, bir verinin veri tabanı tablosundaki bir sütun değerinin, başka bir tablonun bir sütun değeri ile karşılıklı olarak ilişkili olması durumudur. İş tanım ya da güvenlik gereksinim dokümanlarında aksi belirtilmedikçe, verinin bütünlüğünün korunabilmesi için, birbirine bağımlı varlıkların veri tutarlılık kontrollerinin uygulama içinde değil de verinin saklandığı veri tabanında sağlanması gerekmektedir. Bu sayede veri başka ortamlara taşınsa bile bütünlüğü sağlanmış olacaktır.

Referans:

- http://msdn.microsoft.com/en-us/library/aa292166(v=vs.71).aspx

1.16 Uygulama geri dönüşü olmayacak şekilde silinmiş verilerin artık uygulama içinde erişilememesini ve yeni üretilen verilerin, geri dönüşü olmayacak şekilde silinmiş bilgi içermemesini garantiye etmelidir.

Etiket: Veri Bütünlüğü, Veri silme

Neden: Saldırganlar, sadece mantıksal olarak silinmiş ancak geri döndürülebilecek verileri geri getirerek bunlardan istifade etmeye çalışacaklardır.

Nasıl: Bknz: Madde 1.8 ve 1.15

1.17 Son kullanıcı, uygulamayı birden fazla ekranda kullanıyorsa bu ekranlar ya da raporlama ara yüzleri üzerinden sunulan bilginin tutarlılığını sağlamalıdır.

Etiket: Veri Bütünlüğü, Eş zamanlılık

Neden: Bir ara yüzde değiştirilen veri, uygulamanın diğer ara yüzlerinde aynı anda güncellenmezse, güvenlik kontrollerinde karışıklıklara neden olacak problemler ortaya çıkacaktır; Veri değişikliklerinin sadece uygulama özelliği olarak mı, yoksa bir saldırıdan dolayı mı tutarsız olduğu bilinemeyecektir.

Nasıl: Aynı mantıksal veri(logical data) ya da bilgi farklı istemcilere aynı anda, aynı içerikle gösterilmelidir. Aynı veri üzerinde aynı anda birden fazla güncelleme yapılıyor ise güncellemelerin tutarlılığının sağlayacağı mekanizmalar geliştirilmelidir ya da kullanılmalıdır. Özellikle istemciye verinin doğrudan gönderilmesi durumlarında (Push metodu) bu duruma dikkat edilmelidir.

2. KİMLİK DOĞRULAMA

2.1 Uygulama, kimlik doğrulama yapmadan, anonim erişime kapalı olan servislerine/ara-yüzlerine/sayfalarına erişimi engellemelidir ya da bu tür erişimlerin denetlenebileceği ve yönetilebileceği mekanizmalar sunmalıdır.

Etiket: Kimlik Doğrulama, Erişim Kontrolü, Ara yüz

Neden: Kimlik doğrulamanın olmadığı uygulama birimleri güvenlik zafiyeti oluştururlar.

Nasıl: Hassas verilerin işlendiği uygulama birimlerinde kimlik doğrulama teknolojileri kullanılmalıdır.

Referans:

- http://www.bilgiguvenligi.gov.tr/kimlik-yonetimi/kimlik-dogrulama-faktorler-ve-bilesenleri.html

2.2 Uygulamanın, müşteri tarafından tanımlanmış olan kritik servisleri ve ayarları için 2 veya 3 faktörlü kimlik doğrulama kullanılmalıdır.

Etiket: Kimlik Doğrulama

Neden: Son kullanıcılar, her ne kadar parola politikaları ile güvenli olmaya zorlansalar bile, parola seçerken kolaylığı güvenliğin önüne koymaya eğimlidirler. Bu yüzden saldırganlar tek faktörlü kimlik doğrulama yapan uygulamaları ele geçirmekte daha başarılı olmaktadırlar.

Nasıl: Gerekli görülen durumlarda kullanıcı, bildiği parolaya ek olarak sahip olduğu(örn. akıllı kart) veya kullanıcıya ait(örn. parmak izi) bir kimlik doğrulama metodunu da kullanmalıdır.

Güvenlik tasarım dokümanında ve risk analizinde 2 ya da 3 faktör kimlik doğrulamanın riski nasıl aza indirgediğine yer verilmelidir.

Referans:

- http://www.bilgiguvenligi.gov.tr/kimlik-yonetimi/iki-faktorlu-dogrulama-2-factor-authentication-3.html

2.3 Uygulama, önceden belirlenmiş sayıda yanlış kimlik doğrulama denemesinden sonra kullanıcıdan CAPTCHA* talep etmelidir. DOS ya da kaba kuvvet saldırısı (bruteforce) ihtimalinin olmadığı ortamlarda, önceden belirlenmiş hatalı kimlik doğrulama denemesinin ardından hesap kitlenmelidir. Uygulama hesabı tekrar aktive etmek için bir mekanizma sunulmalıdır.

Etiket: Kimlik doğrulama, DOS Belirleme

Neden: Saldırganlar, betikler ve zararlı yazılımlar yardımı ile çok kısa zamanda birçok parolayı deneyerek uygulamanın kimlik doğrulama mekanizmalarını baypas etmeye çalışmaktadırlar. Bunun önüne geçmek için parolayı sağlayanın insan mı yoksa makine mi olduğu ayrımını yapmak gerekir.

Nasıl: Kullanıcı yazılımda kimlik doğrularken, belirtilen resimlerde yer alan yazıyı girmesi istenmektedir. Bu şekilde, istek gönderenin gerçekten bir insan mı yoksa bir yazılım mı olduğu tespit edilir.

Referans:

- http://www.bilgiguvenligi.gov.tr/teknik-yazilar-kategorisi/captcha-kullanimi.html
- http://recaptcha.net/captcha.html

2.4 Kimlik doğrulama esnasında ve hata durumlarında uygulama, kullanıcıya mümkün olduğunca az bilgi ile geri-bildirim yapmalıdır.

Etiket: Kimlik Doğrulama, Hata Bildirimi

Neden: Bu önlemin amacı, saldırganların uygulamanın kimlik doğrulama prosedürü hakkında, işlerine yarayabilecek bilgi edinmelerini engellemektir.

Nasıl: Örneğin, hatalı kimlik doğrulama denemesi sonrası kullanıcıya sadece kimlik doğrulama işleminde hata oluştuğu mesajı verilmelidir. Hatanın kaynağı ve sebebi kullanıcıya bildirilmemelidir. Daha önceden belirlenen hata kodları kullanılabilir. Örneğin;

> "Hata0001: Kimlik doğrulama işlemi başarısız. Sistem yöneticinizle görüşünüz. "

Referans:

- https://www.bilgiguvenligi.gov.tr/web-guvenligi/ozel-yapim-hata-sayfalari-custom-error-pages.html

- https://www.owasp.org/index.php/Error_Handling,_Auditing_and_Logging

2.5 Uygulama, kimlik doğrulamaları yapılmış kullanıcılara kendi parolalarını ve güvenlik kontrolleri için kullanılan profil bilgilerini (Örn. e-mail adresi) değiştirme imkanı vermeli

Etiket: Güvenlik Yönetimi, Kimlik Doğrulama

Neden: Kullanıcının kimlik bilgileri zamanla değişmiş olabilir.

Nasıl: Kullanıcılara, kendi kimlik bilgilerini yönetmek için ara yüz sunulmalıdır.

2.6 Uygulama, kullanıcıya parola kurtarma (password recovery) mekanizmaları sunmalıdır. Parola kurtarma fonksiyonu güvenlik zafiyeti içermeyecek şekilde gerçeklenmelidir.

Etiket: Güvenlik Yönetimi, Kimlik Doğrulama

Neden: Kullanıcılar, parolalarını kaybetmiş veya unutmuş olabilirler.

Nasıl: Parolalar, kullanıcının daha önce belirledikleri güvenli bir ortama(örn. email adresleri, cep telefonları veya ev adresleri) gönderilebilir.

2.7 Uygulama kullanıcısının parolasını unuttuğu veya başka bir nedenle mevcut parola ile uygulamada oturum açamadığı durumlar için kullanıcıya parolayı sıfırlayabileceği bir ekran veya fonksiyon sunmalıdır.

Etiket: Güvenlik Yönetimi, Kimlik Doğrulama

Neden: Parola yönetimi, uygulama güvenliğinin önemli bir parçasıdır. Kullanıcın, unuttuğu veya kaybettiği parolasını, diğer insanlarla en az etkileşimle tekrar edinmesini sağlamak gerekmektedir.

Nasıl: Parola kurtarma işlemi için; sadece kullanıcının sağlayabileceği ek bilgilerin sorulması, akıllı kart üzerinden kimliğin tespiti ya da e-posta adresine gönderilen tekilliği garantilenmiş bir bağlantı üzerinden parola kurtarma fonksiyonu sunulabilir. Parola kurtarma işlemi kullanıcının kendi parolasını belirlemesine izin vermelidir.

2.8 Kullanıcıya uygulamaya erişebilmek için otomatik olarak verilen ilk parola, saldırılara dirençli, benzersiz ve sınırlı geçerlilik süresine sahip olmalıdırlar.

Etiket: Parola Politikası, Kimlik Doğrulama

Neden: Saldırganların, bir yazılımı ele geçirmek için ilk kontrol ettikleri şeylerden biri, varsayılan(default) parolalardır.

Nasıl: Kullanıcıya verilen ilk parolalar tahmin edilemeyecek zorlukta olmalı ve kullanıcı bu parolaları hızlı bir şekilde değiştirmeye zorlanmalıdır. Ayrıca, uygulama yöneticileri, uygulamanın güvenlik ayarlarının yapıldığı ayar dosyaları için konulan parolaları da hızlı bir şekilde değiştirmeye zorlanmalıdırlar.

2.9 Uygulama kullanıcıları, parola belirlerken veya mevcut parolalarını değiştirirken GİD'de belirtilmiş parola politikalarına uymaya zorlamalıdır.

Etiket: Parola Politikası, Kimlik Doğrulama

Neden: Kullanıcı seçimlerinden kaynaklanan zayıf parolaların önüne geçmek için parola politikası gereklidir.

Nasıl: Aşağıdaki alt maddelerde parola politikaları listelenmiştir.

2.9.1 Kullanıcılar, otomatik olarak aldıkları parolaları ilk başarılı kimlik doğrulama sonrasında sadece kendi bildikleri parolalarla değiştirmeye zorlanmalıdırlar.

2.9.2 Tüm parolalar önceden belirlenmiş minimum uzunluğa ve maksimum geçerlilik süresine sahip olmadırlar.

2.9.3 Parolalar sadece alfa-numerik karakterler içermelidir ve uygulama tarafından bu kural zorlanmalıdır.

2.9.4 Parolalar sözlük saldırılarına (dictionary attack) karşı dayanıklı olmalıdırlar. Parolaların zorluk dereceleri uygulama tarafından kontrol edilmeli ve tahmin edilebilir parolalar için son kullanıcı uyarılmalıdır.

2.9.5 Parolalarda aynı karakterler belirlenen bir sayıdan fazla tekrarlanmamalıdırlar.

2.9.6 Parolalar önceden belirlenmiş bir süre sonunda değiştirilmelidirler. Standart uygulama kullanıcıları için tavsiye edilen süre 180 gündür. Yönetici yetkili kullanıcıların parolaları daha sık aralıklarla değiştirilmelidir.

2.9.7 Belli sayıdaki önceden kullanılmış parolaların tekrar kullanılması kısıtlanmalıdır.

2.9.8 Uygulama parola değiştirme fonksiyonları içermelidir. Parolayı değiştirebilmek için kullanıcı eski ve yeni parolaları sağlamalıdır.

2.9.9 Parola politikasının uygulama yöneticisi tarafından değiştirilmesine imkân sağlayacak uygulama ara yüzleri sunulmalıdır.

Referans:

- http://www.bilgiguvenligi.gov.tr/web-guvenligi/web-uygulamalarinda-sifre-guvenligi.html

Kimlik Doğrulama Teknikleri

2.10 En iyi uygulama tavsiyesi olarak (best practice) önerilen "autocomplete=off" bayrağı kullanılarak internet tarayıcılarının kimlik bilgilerini tutmasının engellenmesidir.

Etiket: Kimlik Doğrulama, Gizlilik

Neden: Saldırganlar, kullanıcıların, her seferinde tekrar girmek istemedikleri kimlik bilgilerini kaydettikleri varsayımıyla tarayıcının parola tutulan dosyalarını(Örn. chrome://settings/passwords) kontrol etmektedirler.

Nasıl: Bu işlem doğrudan sayfa içinde

```
Form.setAttribute( "autocomplete", "off" );
FormElemani.setAttribute( "autocomplete", "off" );
```

şeklinde JavaScript komutları kullanılarak gerçekleştirilebilir. "autocomplete" bayrağı W3C tarafından tanımlanmış bir bayrak olmadığından HTML form alanları içinde doğrudan kullanılması önerilmemektedir. HTML5 ile birlikte bu bayrak form alanları için de kullanılabilir.

2.11 Kimlik doğrulama, parola sıfırlama vs. işlemleri sonrası uygulamanın başka bir sayfasına internet tarayıcının yönlendirilmesini(redirect) sağlamak amacıyla HTML "HTTP-EQUIV=REFRESH " kullanmaktan kaçınılmalıdır.

Etiket: Kimlik Doğrulama

Neden: Başka bir sayfaya geçmek istendiğinde kullanılan http-refresh metodu, geçilen sayfa ile önceki sayfa arasındaki bağı koparmayacaktır. Yani saldırgan "geri" düğmesi ile dolu kimlik doğrulama form alanlarının elde edebilecektir.

Nasıl: Örneğin http://www.sirket.com.tr/login.php sayfasında kimlik doğrulandıktan sonra http://www.sirket.com.tr ana sayfasına son kullanıcıyı yönlendirmek için

```
<META HTTP-EQUIV=REFRESH CONTENT="1; URL=http://www.sirket.com.tr">
```

metodunu kullanmak "geri" düğmesi kullanılarak daha önce gönderilen POST verisinin tekrar gönderilmesini (dolayısıyla kimlik doğrulama bilgilerine aynı bilgisayarda başka bir kullanıcının erişimini) sağlayacağından sakıncalıdır. Bunun yerine "HTTP Redirect" (HTTP 302 kodu ile yönlendirme) kullanılmalıdır.

Referans:

- https://www.owasp.org/index.php/Top_10_2010-A10-Unvalidated_Redirects_and_Forwards

2.12 Uygulama tarafından bırakılan çerez(cookie)* için "httponly" aktive edilmelidir. Buna ek olarak, HTTPS protokolü kullanılan bağlantılarda çerezler için "secure" parametresi aktif olmalıdır.

Etiket: Kimlik Doğrulama, Güvenli Veri Taşıma

Neden: XSS açıklarından yararlanan saldırganlar, JavaScript'in çerezleri yönetmeyi sağlayan özelliğinden yararlanarak sayfayı gezen kullanıcının çerez bilgilerini alıp, sisteme o ziyaretçiymiş gibi girebilirler.

Nasıl: HttpOnly olarak işaretlenen çerezlere tarayıcı, javascript komutlarıyla erişemez. Bu da çerezleri XSS* saldırısına karşı korumaktadır. HTTP başlıklarındaki Set-Cookie özelliğine HttpOnly deyimi eklenmelidir. Bu şekilde destekleyen tarayıcılar, HttpOnly deyiminin geçtiği çerezi JavaScript yoluyla alamayacaktır.

Ayrıca HTTPS üzerinden yollanan çerezler eğer "Secure" olarak işaretlenmez ise HTTP üzerinden şifresiz olarak da gönderilirler. Bütün bir uygulama boyunca oturum işlemlerinin geçtiği her adımda HTTPS kullanılan bir sistemde çerezleri "Secure" olarak işaretlememek, bir saldırganının oturum bilgisini HTTP üzerinden elde etmesi ile sonuçlanabilir.

Eğer bütün uygulama kapsamında HTTPS üzerinden şifreli bir bağlantı gerçekleştiriliyorsa, HTTP başlıklarındaki Set-Cookie özelliğine Secure deyimi eklenmelidir.

Referans:

- http://www.bilgiguvenligi.gov.tr/siniflandirilmamis/oturum-acma-sistemleri-tasariminda-guvenlik.html
- https://www.owasp.org/index.php/HttpOnly

2.13 Uygulamanın, kimlik doğrulamada Tek Giriş*(Single Sign-On)(SSO) teknolojilerini kullanması ön görülüyorsa, uygulamaya özel Tek Giriş mekanizması yazmaktan ziyade, piyasada hali hazırda bir kısmı açık kaynaklı olarak sunuluna Tek Giriş teknolojilerini kullanmak güvenlik açısından daha uygun olacaktır.

Etiket: Tek giriş, Kimlik Doğrulama

Neden: Uygulamaya özel yazılmış Tek Giriş çözümleri istimara maruz kalabilecek açıklıklar içerebilir.

Nasıl: Kullanılabilecek Tek Giriş Mekanizmalarına örnekler;

OAuth: Open Authorization sunucu tarafından üretilen bir jetonu(token) kullanarak bir uygulamayı veya servisi jetonu üreten uygulamanın sunucusuna kullanıcı olarak tanıtan açık kaynaklı bir protokoldür. Bu teknoloji kullanılacaksa OAuth 2.0 versiyonu kullanılmalıdır. OAuth 2.0 Google, Facebook, Twitter ve Microsoft'un API'leri tarafından kullanılmaktadır.

OpenID: http bazlı bir protokoldür. Bir kimlik sağlayıcının(Identity provider) başlattığı Single Sign-on(SSO)'u kullanarak, bu servis sağlayıcısına güvenen farklı uygulama ve servislere tekrardan kimlik doğrulama ihtiyacı olmadan bağlanmayı sağlar. Google, Facebook, Twitter, Stack Exchange ve Yahoo çok bilinen kimlik sağlayıcılarındandırlar. Kurumsal olmayan uygulama ve çevreler için OpenID güvenli ve kolay bir kimlik doğrulama çözümü olarak görülebilir.

SAML: Security Assertion Markup Language(SAML), OpenID'nin rakibi gibi görülse de önemli farkları vardır. Güvenlik açısından istikrarlı 2.0 versiyonunun kullanılması tavsiye olunur. SAML OpenID'ye benzer şekilde güvenilir kimlik sağlayıcıları kullanır ancak OpenID'den farkı XML bazlı olması ve çok saha esnek bir yapıda olmasıdır; XML verileri göndererek tarayıcı yönlendirmeleri üzerine bina edilmiştir. OpenID'nin yaygınlığına karşın, SAML kurumsal çözümler için yoğun olarak kullanılmaktadır. SAML'ın intranet içinde bile(içeriden bir sunucuyu kimlik sağlayıcı olarak kullanarak) kullanıldığı birçok senaryo görülebilir. Kurumsal çözümlerinde web uygulamaları ve servislerinin gerektiği durumlarda SSO gerçeklemeleri için SAP ERP*, ve SharePoint* de SAML 2.0 kullanmaya karar vermişlerdir.

Referans:

- http://www.bilgiguvenligi.gov.tr/kimlik-yonetimi/uygulamalarin-opensso-tabanli-kimlik-yonetim-sistemi-entegrasyonu.html
- https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
- http://oauth.net/about/
- http://openid.net/get-an-openid/what-is-openid/

Oturum Yönetimi

2.14 Uygulama, oturum tekil tanımlayıcısını(Session ID)* korumalıdır.

Etiket: Oturum Yönetimi, oturum kimliği

Neden: Saldırganlar, ele geçirdikleri tekil tanımlayıcılarla, oturum sabitleme(session fixation)* saldırıları gerçekleştirebilirler.

Nasıl: Örneğin güvenlik tasarım ya da iş tanım dokümanlarında aksi belirtilmemişse uygulama kullanıcının tüm oturumunu SSL yoluyla koruyabilir. Bu şekilde oturum kimliğinin network üzerinden okunamaması sağlanmış olacaktır.

2.15 Oturum tekil tanımlayıcısı (Session ID) URL'de gönderilmemeli veya referrer başlığı* içine dâhil edilmemelidir.

Etiket: Oturum Yönetimi, Oturum Kimliği

Neden: Http GET/POST değişkenlerini* veya referrer başlıklarını bir link veya form içinde değiştirerek oturum sabitleme saldırısını gerçekleştirmek çok kolaydır.

Nasıl: Tekil tanımlıyıcıların URL içinde veya POST değişkeni olarak kabul edilmemeleri ile oturum kimliklerinin tarayıcı tarafından ön belleğe alınmasının(cache) önüne geçilmiş olunur. Uygulama kimlik sınama ve oturum bilgilerini GET talebi parçası olarak göndermemelidir.

Bu tür bir ihtiyaç genelde sistemde bulunan farklı uygulamaların (COTS* ya da proje kapsamında kullanılma sunulmuş) aynı oturum bilgisini kullanması durumunda oluşmaktadır. Farklı uygulamaların aynı oturumu kullanması tekil oturum (single sign-on) yöntemi ile sağlanmalıdır.

Referans:

- https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Session_ID_Life_Cycle

2.16 Oturum yönetimi için kullanılan ve uygulamayı kullanan bütün kullanıcılar için tekil olması gereken değerler (session id, token v.b.) güçlü bir rastgele veri üretecinden temin edilmeli ve tahmin edilemez derecede karmaşık olmalıdır.

Etiket: Oturum Yönetimi, Oturum Kimliği

Neden: Eğer bahsi geçen tekil değerler, rastgele değil de bir düzen bağlı olarak hesaplanıyorsa, saldırganlar tek bir tekil değerden yola çıkarak, birçok kullanıcının oturumuna ulaşabilirler.

Nasıl: Kullanıcının kendisi tarafından seçilmiş oturum kimlikleri kabul edilmemelidir. Örneğin Universally Unique Identifier (UUID) bu amaç için kullanılabilir. Farklı yazılım dilleri için UUID oluşturan fonksiyonlar bulunmaktadır.

Referans:

- *https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Session_ID_Entropy*

2.17 Belirlenen süre boyunca aktif olmayan oturumlar otomatik olarak kapatılmalıdır. Uygulama, sistem yöneticilerine bu süreyi ayarlayabilecekleri bir ekran sunmalıdır.

Etiket: Oturum Yönetimi, Oturum Zaman Aşımı

Neden: Oturumunu kapatmayan kullanıcının oturumu, saldırganlar tarafından istismar edilebilir.

Nasıl: Çözüm seçenekleri aşağıda alt maddeler halinde sunulmuştur;

2.17.1 Oturumun bitmesi sonrası kullanıcıya uyarı ekranı gösterilmeli. Kullanıcı, uyarı ekranını belirtilen bir süre (Uygulamanın güvenlik hassasiyetine göre belirlenir) içinde yanıtlamaz ise oturum sonlandırılmalı ve oturuma ait bütün veriler silinmelidir.

2.17.2 Uygulama kullanıcının istediği zaman oturumu kapatmasına izin vermelidir. Diğer bir ifadeyle oturum sonlandırma "Logout" fonksiyonu sunulmalıdır. Herhangi bir oturum oluşturmadan çalışan uygulamalar için (Örn. basit kimlik doğrulama ya da NTLM kimlik doğrulama için çalışan) de internet tarayıcısının kapatılması ile birlikte oturumun sonlandırılacağı uygulamanın uygun bir yerinde belirtilebilir.

2.17.3 Kullanıcının uygulamada açabileceği oturum sayısı sınırlandırılmalı ve uygulama sistem yöneticisine bu sınırı ayarlayabilme imkânı sunmalıdır. Bu özellik hatalı ya da kasıtlı olarak kaynakların tüketilmesine neden olabilecek sayıda oturum oluşturulmasına engel olacak ve diğer kullanıcıların da uygulamayı kullanabilmelerini sağlayacaktır.

2.17.4 Uygulamalarda başarılı kimlik doğrulama ve tekrarlayan kimlik doğrulama (re-authentication) neticesinde her zaman yeni bir oturum bilgisi oluşturulmalıdır. Çıkış işleminden sonra da var olan oturum bilgisi geçersizleştirilmelidir.

3. YETKİLENDİRME

3.1 Uygulama, kullanıcının sadece bilmesi gereken bilgilere erişim sağlayabilecekleri şekilde kısıtlamalar getiren mekanizmalar barındırmalıdır.

Etiket: Yetkilendirme, Gizlilik, Erişim Matrisi

Neden: Hassas bilgilere kimlerin erişebileceğinin ön görülebilmesi ve gizli verilerin kötü niyetli kişilerden korunması için yetkilendirme gerekmektedir.

Nasıl: Uygulama iş nesnelerini bilgi varlığı olarak tanımlanabilir ve her bir bilgi varlığı için yetki ve roller oluşturularak erişim kontrol (yetkilendirme) matrisi oluşturulabilir. Erişim kontrol matrisi* geliştiricilerin uygulama fonksiyonlarını doğru şekilde yetkilendirmeye tabi tutmasını sağlamak açısından da önemlidir.

Örneğin bir bankacılık uygulamasında son kullanıcının döviz hesabı oluşturma izni bulunmuyor ise normal bir kullanıcı yetkileri ile döviz hesabı oluşturma fonksiyona erişimin kısıtlanması gerektiği erişim kontrol matrisinde yer almalıdır. Bu kapsamda döviz hesabı bir iş nesnesi olarak değerlendirilmekte ve bu iş nesnesi için yetkilendirme mekanizması olarak rol yönetim ara yüzünde bu fonksiyonu çalıştırabilecek rollere yetki atamasının yapılmasının sağlanması öngörülebilir.

Uygulama sistem yöneticilerine (yetkili kişi) iş nesneleri, kullanıcı tabloları dışındaki veriler seviyesinde yetki tanımlamaya izin vermelidir.

Referans:

- https://www.bilgiguvenligi.gov.tr/dokuman-yukle/bgys/uekae-bgys-0006-erisim-kontrol-politikasi-olusturma-kilavuzu/download.html
- https://www.owasp.org/index.php/Guide_to_Authorization

3.2 Yetkilendirme sunucu tarafında yapılmalıdır. Sadece istemci tarafında yetkilendirme kontrolü yapılmamalıdır.

Etiket: Yetkilendirme

Neden: İstemci taraflı doğrulamalar (Örn. JavaScript ile form alanlarında kural dışı veri girişi doğrulaması) hatalı ya da kritik işlem (veri silmesi durumunda uyarı gösterilmesi), istemci tarafı bilgi toplama gibi durumlarda geri bildirim yapmak için kullanılan ve web sunucuya az yük

bindiren kontrollerdir. Ancak istemci taraflı doğrulamalar aşılabilir, atlatılabilir. Saldırgan (son kullanıcı da olabilir) sunucu ile istemci arasına girerek sunucudan gönderilen yetkilendirme verilerini manipüle etmek suretiyle hak yükseltme saldırıları gerçekleştirebilir.

Nasıl: Özellikle JavaScript, Java Applet, ActiveX, Flash ve Silverlight gibi istemci tarafı zengin içerik sunan uygulama bileşenleri bu durumdan etkilenmektedir. Genelde yapılan hata ise son kullanıcıya ait yetki bilgilerinin (Örn. claim) istemci tarafı bileşende tutulması ve bu bilgilere göre yetkilendirme kararlarının istemci tarafında verilmesidir. Bu sayede her kontrol için sunucuya talep gönderilmesine gerek kalmamakta fakat güvenlik zafiyeti oluşturulmaktadır.

Bu durum aynı zamanda hassas verilerin istemci tarafında depolanması olarak da değerlendirilebilir ve zafiyet oluşturmaktadır.

Referans:

- https://www.bilgiguvenligi.gov.tr/kimlik-yonetimi/yetkilendirme-ve-kimlik-yasam-dongusu-baglantisi.html
- https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

3.3 Yetkilendirme yaparken "Rol bazlı" yetkilendirme tavsiye edilmektedir.

Etiket: Yetkilendirme, Rol Bazlı Yetkilendirme

Neden: Rol bazlı yetkilendirme, yönetimi kolay, izlenmesi rahat ve diğer sistemlere göre daha performanslı sonuçlar veren bir yetkilendirme yöntemidir.

Nasıl: Rol bazlı yetkilendirme kuralları aşağıdaki alt başlıklarda açıklanmışlardır:

3.3.1 Uygulamanın iş gereği oluşturacağı rollere ek olarak kurum ya da şirketin organizasyon yapısı da rol tanımlarında kullanılabilir.

3.3.2 Yetkilendirme kapsamına aşağıdaki nesneler(veriler) alınabilir:

- İşletim sistemi üzerinde bulunan ve uygulama tarafından oluşturulan erişilen dosyalar, dizinler
- Dinamik içerik sağlayan sayfalar
- İstemci tarafından gönderilen POST ve GET parametreleri, HTTP başlık (header)* alanları
- Web servisleri ve web servisinin sunduğu fonksiyonlar (operations)
- Statik içerik (HTML, PDF, veritabanı yedek dosyaları v.s.)
- Bilgi varlıkları ve iş nesneleri
- Yazılım nesneleri (object, class tanımları)
- Yazılım nesne metodları (function, procedure vs.)
- Önemli fonksiyon icra eden kod parçacıkları

3.3.3 Kullanıcının rolü, kullanıcıya atanmış fonksiyonları ve nesne tiplerini (sayfa, servis…) belirlemelidir. Herhangi bir kullanıcının ya da rolün herhangi bir anda ne tür yetkilere sahip olduğu (erişebildiği sayfalar, iş nesneleri ve iş fonksiyonları ve bunlara erişim seviyesi vs.) raporlanabilmelidir.

3.3.4 Kullanıcı sadece yetkilendirildiği rol kapsamında nesnelere(veri) erişebilir. Bu kuralın gerçeklenmesi için rastgele seçilecek birkaç iş fonksiyonu için kontroller (test case) oluşturulmalı ve teste tabi tutulmalıdır.

Referans:

- http://www.bilgiguvenligi.gov.tr/kimlik-yonetimi/etkin-kimlik-yonetim-modeli.html
- https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

3.4 Uygulamada, kullanıcıların yetkilerinin sistem yöneticisi ya da yetkilendirilmiş kişiler tarafından ayarlanabildiği kimlik yönetimi ekranı olmalıdır.

Etiket: Yetkilendirme, Güvenlik Yönetimi

Neden: Kullanışlı bir ayar ve yönetim ara yüzü olmayan yetkilendirme sistemlerinde gözden kaçacak yetki ve rol atamaları olacaktır. Bu durum büyük güvenlik zafiyetlerine yol açabilir. Ayrıca herhangi bir anda raporlama amacıyla hangi kullanıcıların ne tür yetkilere sahip olduğunu görebilmek de önemlidir. Kimlik yönetim ekranı sunmayan uygulamalarda kullanıcı hesabı oluşturma ve yetkilendirme sistem yöneticileri tarafından sağlanmakta ve bu da suistimallere ve hatalara açık olabilmektedir.

Nasıl: Uygulama yöneticileri yetkilendirme ve erişim yönetimi ekranında kullanıcı yönetimi, rol yönetimi ve kullanıcı rol atamalarını yapabilmelidirler.

3.5 Uygulamada kimlik yönetim ekranlarında belirlenen kullanıcılar ve yetkiler dışında yetkilendirme olmamalıdır.

Etiket: Yetkilendirme, Güvenlik Yönetimi

Neden: Tanımlanmamış yetkilendirme kontrolleri güvenlik zafiyeti oluşturacaktır.

Nasıl: Rol bazlı yetkilendirmenin kullanılması durumunda uygulama geliştiricilerin sadece tanımlı rolleri kullanmaları sağlanmalı ve tanımlanmamış yetkilendirme kontrolleri yapmaları engellenmelidir. Örn. izlenebilirlik ve zafiyetlerin önceden engellenmesi açısından kimlik yönetim ekranında tanımlanmamış rollerin kod içinde kullanılmasına izin verilmemelidir.

3.6 Uygulama dokümante edilmemiş ve sistemin çalışmasını etkileyebilecek parametreleri ya da kullanıcı hesaplarını içermemelidir.

Etiket: Kullanıcı hesapları, Güvenlik Yönetimi

Neden: Saldırganlar, dokümante edilmemiş kullanıcı hesapları, uygulama içinde çok rahat bir şekilde bulup istismar edilebilmektedirler.

Nasıl: Kurulumla birlikte gelen varsayılan(default) kullanıcı hesapları ve roller gerekmiyorsa silinmelidir.

3.7 Güvenlik fonksiyonları ile alakalı görüntüleme ve yapılandırma ara yüzlerine/sayfalarına/servislerine sadece güvenlikten sorumlu ve yetkilendirilmiş hesaplar erişebilmelidir.

Etiket: Yetkilendirme, Güvenlik Yönetimi

Neden: Bu kısıtın olmadığı durumlarda, saldırganlar hak yükseltme saldırıları ile güvenlik ayarlarını değiştirebileceklerdir.

Nasıl: Uygulamanın devreye alınımında yukarıda ifade edilen ayar varsayılan olarak yapılmış olmalıdır.

3.8 Her bir İş nesnesi(business object)* için read/write/modify/delete gibi yetkiler tanımlanmalıdır.

Etiket: Yetkilendirme

Neden: Yetkilendirme noktasında en iyi kullanım yukardaki şekildedir. Ayrıca bu ayrıştırma sayesinde alt yetkilere göre kayıt izleme gerçekleştirilebilir.

Nasıl: Bu yetkiler, rol olarak iş nesnesi seviyesinde (örneğin kullanıcı profil bilgileri, ürün bilgileri, vs.) belirlenebilmeli ve gruplanabilmelidir.

3.9 Kimlik yönetim servisleri(kimlik doğrulama (authentication), yetkilendirme (authorisation) ve kimlik veri depoları (user data store) merkezileştirilmelidir.

Etiket: Yetkilendirme, Kimlik Doğrulama, Güvenlik Yönetimi

Neden: Hem kimlik doğrulama, hem de yetkilendirme farklı bileşenler tarafından yapıldığında yönetim karmaşıklığı nedeniyle güvelik zafiyetleri oluşabilmektedir.

Nasıl: Teknik olarak mümkünse sistem kapsamındaki bütün uygulamalar merkezi kimlik yönetim servislerini kullanmalıdır. Kimlik deposu olarak bir merkezi dizin yapısı kullanılabilir.

3.10 Yetkilendirme dinamik olmalı ve yetki kaldırıldığında nesneye ya da servislere erişim mümkün olmamalıdır.

Etiket: Yetkilendirme

Neden: Yetkilendirmeyi, uygulamanın tüm birimlerinde, her bir nesne için statik olarak ayarlamak, performans açısından daha iyi olsa da, her bir ayardan sonra sistemi yeniden başlatmak gerekeceğinden, sürdürülebilir değildir.

Nasıl: Dinamik yetkilendirme, statik yetkilendirmenin aksine, uygulama sunucusundaki XML dosyası ayarları ile değil, veri tabanı veya aktif dizin üzerinde yapılır ve bu yetkiler uygulamada ön belleğe alınıp işletilir. Bunun yönetimi için geliştirilmiş merkezi kimlik doğrulama ve yetkilendirme teknolojileri de kullanılabilir; Örn. .Net Role Provider, Hibernate, JAAS, Apache Shiro vs.

3.11 Kullanıcı organizasyondan ayrıldığında erişim izinleri ve yetkileri kaldırılmalıdır.

Etiket: Yetkilendirme, Güvenlik Yönetimi

Neden: Uygulamanın çalıştığı organizasyondan herhangi bir nedenle ayrılmış kullanıcıların uygulamaya erişimi risk oluşturmaktadır.

Nasıl: Kurumsal bir uygulamada işten ayrılma sürecini destekleyecek şeklide kullanıcı hesaplarını kilitleyecek ve silecek şeklide servisler sunulmalıdır ve bu servisler güvenlik fonksiyonu olarak değerlendirilmelidir; uygulama güvenlik yöneticisi rolüne atanmalıdır.

3.12 Veriye ya da nesnelere erişim imkânı sunan bütün yollar erişim denetimine tabii tutulmalıdır.

Etiket: Yetkilendirme, Erişim Kontrolü

Neden: Sadece veri ve nesnelerin yetkilendirmeye tabi olması yetmeyebilir. Bu veri ve nesnelere ulaşmak için kullanılan dosya ve dizinlere erişim de yetkilendirilmelidir. Bu şekilde güvenliği sağlamlaştırmanın yanında izlenebilirlik de kuvvetlendirilmiş olacaktır.

Nasıl: Örneğin uygulama verilerini bir veri tabanında saklıyor ise veri tabanına erişim yolları ve yöntemleri de güvenlik kontrolüne tabi tutulmalıdır. Aynı şekilde uygulamanın çalıştığı işletim sisteminde bulunan kritik dosyalar (parola içeren dosyalar ya da şifreleme için kullanılan gizli anahtar vs.) ve bunlara erişim yol, yöntem ve araçların kullanımı (örneğin veri tabanı yönetim araçları ya da işletim sistemi yedekleri) da bu kapsamda değerlendirilmelidir.

3.13 Bir kullanıcının birden fazla rolü var ise oturumu kapatmadan roller arası geçiş yapabilmesi sağlanmalıdır.

Etiket: Yetkilendirme, Kullanışlılık

Neden: Kullanışlı olmayan sistemlerin güvenliğinde de er veya geç insan faktöründen doğan problemler çıkacaktır. Sürekli roller arası geçiş yapmak durumunda kalan kullanıcılar, bir müddet sonra uygun olmayan rollerle işlem yapmaya başlayabilirler.

Nasıl: Kullanıcılar için yazılacak hesap yönetim ara yüzünde roller arası geçiş yapmak için gereken ayarlar yapılabiliyor olmalıdır.

3.14 Uygulama kullanıcıya sadece yetkisi ve izni olan fonksiyonları, servisleri ve nesneleri (sayfa, iş nesnesi, ekran vs.) göstermelidir.

Etiket: Yetkilendirme, Ara yüz

Neden: Bilgi güvenliğindeki, sadece bilinmesi gerekeni bilme(need-to-know) prensibi, kullanıcılardan kaynaklanabilecek potansiyel güvenlik açıklıklarının önüne geçilebilmesi için uygulanmalıdır.

Nasıl: Kullanıcıların, yetkileri olmayan fonksiyon, servis ve nesnelere ulaşımlarını kısıtlama yanında, bunların neler olduğunu öğrenmelerine de izin verilmemelidir.

3.15 İşletim sistemi üzerinde bulunan bir araçla uygulama hesaplarına ait parolalarının doğrudan değiştirilmesini engelleyecek mekanizmalar sunulmalıdır.

Etiket: Erişim Kontrolü

Neden: İşletim sistemiz üzerinde çalışan, uygulamaya müdahale potansiyeli olan araçlar uygulama için güvenlik tehdidi oluşturmaktadırlar.

Nasıl: Uygulama dışında çalışan ve uygulamaya müdahale potansiyeli olan araçlar(örneğin MS SQL veri tabanına erişen Management Studio uygulaması) kaldırılmalı ya da erişim denetimine tabi tutulmalıdır.

3.16 Canlı uygulama ortamı, test ve geliştirme veri tabanlarına (örneğin operasyonel, eğitim, alıştırma veri tabanları) bağlanmamalıdır. Aynı şekilde gerçek veri tabanı asla geliştirme ve test ortamlarında kullanılmamalıdır.

Etiket: Erişim kontrolü, test

Neden: Saldırganlar, canlı ortamdan test ve geliştirme ortamlarına atlayarak, oralardaki güvenlik zafiyetlerinden faydalanmaya çalışacaklardır. Test ve canlı veri tabanı arasında doğrudan bir etkileşime imkan sağlanması durumunda özellikle veri tabanları arası yedekleme, test ya da doğrudan veri iletişimi amaçlı veri tabanı bağlantıları (db link) oluşturulması sıklıkla rastlanılan bir güvenlik zafiyet kaynağıdır. Aynı şekilde hassas verilerin canlı ortamda kaldığını garantilemek için her iki sistemin ayrılmış olması faydalı olacaktır.

Nasıl: Test ortamı test veri tabanında, geliştirme ortamı geliştirme veri tabanında, canlı uygulama da gerçek veri tabanında çalışmalıdır.

3.17 Yığın görevleri (batch jobs)* için yetkilendirme ve izleme yapılmalıdır.

Etiket: İzleme, Yetkilendirme

Neden: Yığın görevleri parola içerebilirler.

Nasıl: Bu görevler parola içerebileceklerinden, erişim denetimine tabi tutulmalıdırlar.

4. ERİŞİLEBİLİRLİK

4.1 Kullanıcı sayısının fazla olduğu ve yoğun olarak kullanılan sistemlerde kimlik yönetim servislerinin yük dengeli (load-balancer)* olarak çalıştırılması öngörülmelidir.

Etiket: Erişilebilirlik, Kimlik Doğrulama

Neden: Dönemsel olarak oluşabilecek aşırı yüklenmeler, işlemlerde gecikmeye ve uygulamanın düşmesine neden olabilir.

Nasıl: Yük dengeleyicisinin düşmemesine(single-point of failure) dikkat edilmelidir. Bu yüzden yük dengeleyici ya default gateway'e ya da erişilebilirliğin çok yüksek olduğu garanti edilebilecek bir birime konulmalıdır.

4.2 Uygulamanın belirlenecek bant genişliği(bandwidth) alt sınırında da çalışabilir olması gerekmektedir.

Etiket: Erişilebilirlik, Bant Genişliği

Neden: Uygulama ve uygulama uç bileşenleri (Örn. uzak ofisler ya da sahada bulunan sensörler vs.) arasında bant genişliğinin kısıtlı olarak sunulabilmesi durumları ortaya çıkabilir.

Nasıl: Uygulamanın düşük bant genişliğinde çalışmasının sağlanması için uygulama yapısının iyi düzenlenmiş olması, http isteklerinin minimize edilmesi ve uygulamanın ön belleğe alınabilir(cache-able) kapasitede olması gerektir.

4.3 Uygulamanın belirlenecek bir paket gecikme süresinde(latency) ya da ondan daha aşağıda çalışması gerekmektedir.

Etiket: Erişilebilirlik

Neden: Uygulama ve uygulama uç bileşenleri (Örn. uzak ofisler ya da sahada bulunan sensörler v.s.) arasında paket gecikmelerinin yaşanması durumu ortaya çıkabilir.

Nasıl: Uygulama kodu geri-dönüşlerden(turn-arounds) arındırılmalıdır. Geri-dönüş, verinin karşı tarafa ulaşana kadar işlemin ileri gitmesinin engellenmesine verilen addır. Bunu engellemek için *pipe-lining* kullanılabilir; yani işlemin devam etmesi için isteğin cevabı beklenmesi gerekmiyorsa, işlem devam ederken isteğin cevabı sonradan toplanır.

4.4 Uygulama, yetkilendirilmiş kullanıcılara tekil bir kullanıcının açabileceği oturum sayısını yönetebilme imkânı sunmalıdır.

Etiket: Erişilebilirlik

Neden: Tekil bir kullanıcının sınırsız sayıda oturum açabilmesi performans problemlerine neden olacaktır.

Nasıl: Veri tabanı seviyesinde bağlantı havuzu(connection pooling)* yöntemi ile uygulama seviyesinde IP kısıtlamasına imkân tanıyan modüller ile tekil kullanıcı oturum sayısı yönetilebilir.

4.5 Uygulama, en iyi bağlantı zamanı ve performans prensiplerini uygulamalıdır.

Etiket: Erişilebilirlik, Performans

Neden: Performans prensipleri dikkate alınmadan geliştirilmiş bir uygulamada erişilebilirlik problemleri ortaya çıkacaktır.

Nasıl: Bağlantı zamanı ve performans prensipleri aşağıdaki alt başlıklarda açıklanmışlardır:

4.5.1 Uygulamaya giriş ya da uygulama ekranlarında verinin gösterilmesi, güncellenmesi vs. gibi işlemler için gerekli en fazla sürenin sınırlandırılmasını(connection time-out) sağlayacak bir mekanizma bulunmalıdır.

4.5.2 Uzun süren ve yüksek işlemci/bellek kullanımına neden olan süreçler, fonksiyonlar, sayfalar, sorgular önceden belirlenmeli ve dokümante edilmelidir. Bu tür işlemlerin çalıştırılması sırasında kullanıcı uyarılmalı ve mümkünse sistemin daha az yoğun olduğu saatlerde çalıştırılması için mekanizmalar sunulmalıdır.

4.5.3 Uygulama, hizmet dışı bırakma (DoS) saldırılarına karşı koruma sağlayan mekanizmalar üretmelidir. Kaynakların kullanıcı ya da uygulama bileşeni bazlı sınırlandırılmasını sağlayan mekanizmalar sunulmalıdır.

4.5.4 İşlem, fonksiyon ya da kullanıcı bazlı kaynak kullanım üst sınırlarının tanımlanabilmesi, yönetilebilmesi ve raporlanabilmesi için mekanizmalar ve ara yüzler sunulmalıdır.

4.5.5 Herhangi bir uygulama hesabının belirlenen bir sürede gönderebilecekleri istek sayısı ya da ağ trafik miktarının sınırlandırılabilmesi için mekanizmalar gerçeklenmelidir. Performans artırımı açısından veri tabanı, LDAP dizini* gibi kaynaklara gereksiz erişimler kısıtlanmalıdır, engellenmelidir.

4.5.6 Performans artırımı açısından uygulamanın statik verileri ve gerekiyorsa akıllı bir yöntem ile dinamik verilerin bir kısmını bellekte tutması tekrar kaynak kullanımına neden olmadan son kullanıcıya göstermesi sağlanabilir.

4.6 Uygulama, iş tanımlama dokümanında ya da güvenlik gereksinimlerinde belirtilmesi durumunda, uygulama ara yüzlerinden işlenen ya da saklanan bütün verilerin yedeklerinin alınabilmesine imkân sağlamalıdır.

Etiket: Erişilebilirlik, Yedekleme

Neden: Uygulamanın düşmesi durumunda veri kaybına uğramamak için yedekleme yapılmalıdır.

Nasıl: Örneğin sistem yöneticisi uygulamanın sunacağı bir yedekleme ara yüzünden hem veri tabanı hem de işletim sistemi üzerinde bulunan dosyaların ve verilerin yedeklerini alabilir ve bu yedekleri sürümlü olarak tutabilir.

4.7 GİD'de tüm verilerin yedeklenmesi ön görülüyorsa, uygulama yedekleme işleminin düzenli olarak yapılabilmesi ve takvimlenebilmesini sağlayan mekanizmalar ile yedekleme işlemlerinin sonlanma durumunu ve hatalarını raporlayabileceği bir ara yüz sunmalıdır.

Etiket: İzleme, Denetim, Yedekleme

Neden: Yedekleme süreci ve yedekleme işleminde ortaya çıkabilecek hatalar izlenmelidir. Uygulamanın hem uygulama sunumcu üzerinde hem de veri tabanı ya da benzeri kaynaklarda veri sakladığını varsayılarak merkezi bir yerden hem dosya sistemi üzerinde hem de veri tabanında bulunan verilerin yedeklerinin alındığını garantilemek önemlidir. Ayrıca merkezi

olarak yönetilmeyen bir yedekleme işlemi, yedekten dönme sırasında zorluk yaşanmasına neden olabilir.

Nasıl: Yedekleme yönetimi ara yüzü yazılıp, yetkili kullanıcıların uygulamanın ne zaman ve hangi aralıklarla yedekleme yapacağını ayarlayabildikleri bir mekanizma oluşturulabilir.

4.8 Uygulama, iş tanımlama dokümanında ya da GİD'de belirtilmesi durumunda, uygulama ara yüzlerinden alınan yedekler üzerinden veri kurtarma işlemini gerçekleştirebilmelidir.

Etiket: Erişilebilirlik, Yedekleme

Neden: Uygulamanın düştüğü durumlarda, uygulama veri kaybının önüne geçmek için veri yedeklerini veri kaybı olan yerlere geri yükleme yapabilmelidir.

Nasıl: Veri kurtarma işlemi için gerekli hakların, bu işlemi gerçekleştirecek proseslere atanmış olmasına dikkat edilmelidir.

4.9 GİD'de yedekleme isteri bulunuyorsa, hem yedekleme hem de yedekten kurtarma işlemleri ve süreçleri dokümante edilmelidir.

Etiket: Yedekleme, dokümantasyon

Neden: Yedekleme ve veri geri yükleme işlemlerinin sağlıklı bir şekilde yapılması ve gerekli durumlarda tekrar edilebilmeleri için bu süreçlerin dokümantasyonunun iyi yapılmış olması gerekmektedir.

Nasıl: Bu kapsamda yedekleme ve kurtarma ara yüzünün kullanımı, yedekleme ve kurtarma işlemlerinin elle yapılabilmesi için yönergeler dokümante edilmelidir. Yedekleme ve kurtarma ek önlemleri olarak ISO IEC 27001:2005 Bölüm A.10.5 doğrultusunda ISO IEC 27002 Bölüm 10.5 altında listelenmiş olan kontroller sağlanabilir.

5. İZLEME VE DENETİM

5.1 İzleme kayıtlarının doğru zaman bilgisi ile oluşturulması sağlanmalıdır.

Etiket: İzleme

Neden: Kayıtların bütünlüğünün garanti altına alınması için zaman bilgisinin doğru olması önemlidir.

Nasıl: Zaman bilgisinin alındığı kaynak uygulama dışında bir sistem bileşeni ise sistem sahibinin bilgilendirilmesini ya da zaman bilgisinin güncel tutulması için uygulamanın koştuğu işletim sistemi üzerinde zaman güncelleme servislerinin çalışması (NTP) sağlanabilir.

5.2 Uygulamanın içinde kayıtları tutulacak güvenlik olayları dokümante edilmiş olmalıdır.

Etiket: İzleme

Neden: İyi dokümente edilmemiş güvenlik olayları, kayıtlarda gözden kaçabilir ve uygulamada kritik açıklıklar ortaya çıkabilir.

Nasıl: Uygulama geliştirme öncesinde GİD'de izlenecek güvenlik olayları belirlenmelidir. Kayıtların ne kadar süre tutulacağı da aynı dokümanda yer almalıdır. ISO IEC 27001:2005 Bölüm A.10.10 doğrultusunda ISO IEC 27002 Bölüm 10.10.3 altında listelenmiş olan kontrollerin GİD'de yer alması ve dolayısıyla gerçeklenmesi sağlanabilir.

5.3 Bir güvenlik ihlali durumunda, kontrol edilmesi gereken güvenlik fonksiyonları ve bu fonksiyonların bulunduğu ekranlar ve sayfalar ya da yapılandırma dosyaları dokümante edilmelidir.

Etiket: İzleme, Denetim

Neden: Güvenlik ihlali durumlarında hızlı tepkiler vermek gerekebilir. Bu durumlarda hangi ekranda, hangi güvenlik fonksiyonunun kontrol edilmesi gerektiğinin dokümante edilmiş olması önemlidir.

Nasıl: Bu güvenlik fonksiyonlarına örnek olarak aşağıda alt maddeler ele alınabilir;

5.3.1 Kimlik doğrulama (Oturum açma kayıtlarının izlenmesi)

5.3.2 Gizlilik ihlali (işletim sisteminde bulunan kritik dosyalara; açık anahtar, parola bulunduran dosyalar vs., veri tabanında saklanan hassas verilere; hesap bilgileri, hassas iş bilgileri vs. kontrolsüz erişim sağlanması durumunun nasıl izleneceği)

5.3.3 Erişilebilirlik kontrolleri (kapanmış olan servisler ve kapanma nedenleri, yedekleme işlemlerinin başarısız olması, uygulamanın yanıt sürelerinin neden arttığı ve en çok kaynak tüketen sayfalar ya da fonksiyonların tespiti)

5.3.4 Yetkilendirme (Yönetici yetkisinin atanması durumlarının izlenmesi, işletim sistemi seviyesinde dosyaların ve dizinlerin yetkilendirmelerin doğruluğu, veri tabanı yetkilendirme kontrollerinin durumu, merkezi kimlik doğrulama ya da yetkilendirme kullanılıyor ise sistem bazında yetkilendirme ve oturum açma girişimlerinin izlenmesi)

5.4 Müşteri talebi doğrultusunda dokümante edilen, güvenlik ihlallerine maruz kalmış fonksiyonlar(Bknz: 5.3) için otomatik müdahale veya alarm mekanizmaları gerçeklenmelidir.

Etiket: İzleme, Denetim

Neden: Güvenlik ihlalleri olaylarına müdahale hızlı olması gerekmektedir. Bu nedenle müdahalenin otomatize olması ve yetkili kullanıcıların ihlallerden hızlı bir şekilde haberdar edilmeleri önemlidir.

Nasıl: Bu mekanizmalara gerçek zamanlı alarmlar, riskli işlemlerin durdurulması, servislerin kapatılması, kullanıcı hesaplarının ağdan çıkarılması veya dondurulması örnek olarak verilebilir.

5.5 Denetlenmesi gereken olayların tümü için denetim kaydı tutulmalıdır. Bu kayıtlar kullanıcı kimliği ile söz konusu olayları birleştirmeli, olayın tarih ve saatini, ne tür bir olay olduğunu, kullanıcı kimliğini ve olayın sonucunu içermelidir.

Etiket: Denetim, İzleme

Neden: Meydana gelen güvenlik ihlallerinin kimden dolayı kaynaklandığını bulmak güvenlik adına önemlidir.

Nasıl: Denetlenmesi ve kayıt tutulması gereken sistem olayları aşağıdaki alt maddelerde sıralanmıştır.

5.5.1 Uygulama sunucu ve/veya bileşenlerinin (modüller, eklentiler vs.) açılması, yeniden başlatılmaları ve kapanışlarını

5.5.2 Oturum açılış (oturum açma girişimleri dâhil) ve kapanışları

5.5.3 Yetki ve rol güncellemeleri ve kullanıcı ya da gruplara rol atamaları

5.5.4 Güvenlik fonksiyonları ile ilgili uygulama ve sistem ayarlarını değiştirme ekranına erişim durumları ve güncellemeler (kayıt ve izleme işlevleri dâhil)

5.5.5 İzleme ve kayıt (audit) fonksiyonlarının açılış ve kapanışlarını

5.5.6 Güvenlik fonksiyonu sunan servislere, sayfalara ya da verilere her türlü erişim (Örn. veri tanında tutulan kullanıcı ve parola tablolarına erişim, şifreleme için kullanılan anahtarlara erişim, kullanıcı parola güncelleme ve rol atama sayfalarına erişim, erişilebilirlik yapılandırmalarını sunan ekranlara ve sayfalara erişim, izleme kayıtlarını içeren ekranlara erişim vs.)

5.5.7 İzleme kayıtlarının oluşturulması, silinmesi veya değiştirilmesi

5.5.8 Sitemin tarih ve saatinin değiştirilmesi

5.5.9 Sistem kaynaklarına başarısız erişim denemeleri

5.6 İzleme kayıtları ara yüzü olay izlemeyi kolaylaştıracak şekilde filtrenebilmeli ve olay ilişkilendirmesi sağlayacak şekilde sorgulara izin vermelidir.

Etiket: İzleme, Kullanışlılık

Neden: İzleme kayıtlarının anlamlandırılması ve gereken düzeltmelerin hızlı bir şekilde yapılabilmesi için kullanışlı bir ara yüz olması gerekir.

Nasıl: Örneğin; kullanıcı adı ve tarihe göre sıralama yapılabilmelidir, filtrelenebilmelidir.

5.7 İzleme kayıtlarının yetkisiz silinmeden ve/veya değiştirilmeden korunması gerekmektedir.

Etiket: İzleme, Yetkilendirme, Veri bütünlüğü

Neden: İzleme kayıtları, bütünlük kontrolüne tabi tutulması gereken hassas veri kategorisindedir.

Nasıl: Madde 1, Verinin Korunması başlığı altındaki maddeler izleme kayıtları için de uygulanmalıdır. Ayrıca izleme (audit) kayıt mekanizmaları sürekli çalışıyor vaziyette olmalıdır.

5.8 İzleme kayıtlarına erişim de, erişim denetimine tabii olmalıdır. Bu bilgilere sadece güvenlik yöneticilerinin erişmeleri sağlanmalıdır.

Etiket: İzleme, Erişim

Neden: İzleme kayıtları, erişim kontrolüne tabi tutulması gereken hassas veri kategorisindedir.

Nasıl: Madde 3, yetkilendirme başlığı altındaki erişim kontrolü hususundaki maddeler izleme kayıtları için de uygulanmalıdır.

5.9 İzleme kayıtlarının arşivlenmesi ve bu arşivlerin bakımı mümkün olmalıdır.

Etiket: İzleme, arşivleme

Neden: İzleme kayıtları, erişim kontrolüne tabi tutulması gereken hassas veri kategorisindedir.

Nasıl: Arşivlenmiş denetim bilgileri bozulmaya ve yetkisiz değiştirilmeye/silinmeye karşı korunmalıdır.

5.10 İzleme kayıtları, güvenlik yöneticisinin belirlediği ya da uygun bir standarda göre belirlenmiş bir süre zarfı müddetince tutulmalıdır.

Etiket: İzleme, Güvenlik Yönetimi

Neden: Uygulamada ortaya çıkması muhtemel güvenlik olaylarının nedenlerinin araştırılması için geriye dönük izleme kayıtları çok önemlidir.

Nasıl: Önerilen süre 5651 yasasına göre en az 6 ay, en fazla 2 yıldır.

5.11 Kritik sistemler ve uygulamalar için ileri seviye güvenlik gerekiyorsa sistem aktivitelerinin ve izleme kayıtlarını otomatik olarak inceleyen ve analiz eden, potansiyel güvenlik ihlallerini arayan bir mekanizma sağlanabilir.

Etiket: İzleme, Denetim

Neden: İleri seviyede güvenlik gerektiren hassas uygulamalarda, her türlü güvenlik olayına çok hızlı müdahale etmek gerekmektedir.

Nasıl: Bu işi yapan üçüncü parti mekanizmalar da kullanılabilir. Örn. Arcsight, SCOM(System Center Operations Manager) vs.

5.12 Tüm hatalar, hata kaydı olarak kaydedilmeli ve bu kayıtlar hem merkezi hem de yerel olarak saklanmalıdır.

Etiket: İzleme, Hata İzleme, Arşivleme

Neden: Özellikle hatalı oturum açma denemeleri, beklenmeyen formatta bir xml dosyası veya HTTP isteğinin alınması durumunda önemlidir.

Nasıl: Merkezi izleme için 3. Parti izleme ürünleri kullanılabilir. Örn. AlienVault OSSIM, HP ArcSight vs.

5.13 Uygulama, yetkilendirilmiş kullanıcının, kullanıcı aktivitelerini ve güvenlik fonksiyonları ile ilgili hata kayıtlarını görebileceği bir ara yüz sunmalıdır.

Etiket: İzleme, Yetkilendirme, Güvenlik Yönetimi

Neden: Özellikle hatalı oturum açma denemeleri, beklenmeyen formatta bir xml dosyası veya HTTP isteğinin alınması durumunda önemlidir.

Nasıl: Merkezi izleme için 3. Parti izleme ürünleri kullanılabilir. Örn. AlienVault OSSIM, HP ArcSight vs.

5.14 Uygulama, yetkilendirilmiş kullanıcının hangi güvenlik kayıtlarının izleme kapsamında değerlendirileceğine karar verebileceği bir ara yüz sunmalıdır.

Etiket: İzleme, Yetkilendirme, Kullanışlılık

Neden: İzleme kayıtları, kendi içindi farklı hassasiyet seviyesinde bilgiler barındırabilir.

Nasıl: Örneğin uygulama sunucusunun açılış ve kapanış izleme kayıtları kapsam harici tutulurken, kullanıcı erişimleri kapsama dâhil edilebilir.

5.15 Seçilmiş bazı izleme kayıtlarının veri ile birlikte tutulması gerekmektedir.

Etiket: İzleme, Veri Bütünlüğü

Neden: Veriler veri tabanında tutulurken, veri oluşturma ve güncelleme işlemleri nihai olarak uygulama üzerinden yapılmaktadır. Bu tür işlemlere ait kayıtların sadece uygulama üzerinde tutulması yeterli değildir. Veriler veri tabanında tutulduğu için verilere ait kayıt bütünlüğü bozulabilir.

Nasıl: Örneğin, bütün veri tabanı tablolarında ek dört adet sütun oluşturulabilir; oluşturma tarihi, oluşturan kişi, güncelleme tarihi ve güncelleyen kişi gibi) ve yeni tablo satırları oluşturulduğunda ya da satırlar güncellendiğinde bu sütunlar da güncellenmelidir. Burada dikkat edilmesi gereken konu "kişi" ile kastedilen veri tabanı kullanıcısı değil uygulama kullanıcısına ait ID ya da benzeri tanımlayıcı bir değerdir.

5.16 GİD'de yedekleme isteri bulunuyorsa, uygulama, izleme kayıtlarının doğru bir şekilde arşivlendiğini garanti etmek üzere bütünlüğü kontrol edebilecek önlemler almalıdır.

Etiket: İzleme, Veri Bütünlüğü, Arşivleme

Neden: Saldırganlar izleme kayıtlarına sadece veri kaydedilirken değil, arşivlenirken de müdahale edebilirler.

Nasıl: İzleme kayıtları arşivlenirken saat başı veya günlük zaman damgalı kriptografik özetleri(hash) de alınıp kaydedilmelidir.

5.17 Uygulama, yetkilendirilmiş kullanıcıların izleme kayıtlarını filtrelemelerine izin verebilmelidir.

Etiket: İzleme

Neden: İzleme kayıtlarını anlamlandırmak ve efektif bir şekilde değerlendirmek için filtreleme gerekmektedir.

Nasıl: Aşağıdaki donelere göre filtreleme yapılabilir.

5.17.1 Tarih/saat aralığı

5.17.2 Olay IDsi

5.17.3 Olay tipi

5.17.4 Kategori

5.17.5 Açıklayıcı yazı

5.17.6 İşletim sistemi kullanıcısı veya grubu veya kaynağı

5.18 **GİD'de yedekleme isteri bulunuyorsa, uygulama, yetkilendirilmiş kullanıcıya izleme kayıtlarını temizleme ve bunların yedeklerini alma kabiliyetleri sunmalıdır.**

Etiket: İzleme, Kullanışlılık

Neden: İzleme kayıtlarını anlamlandırmak ve efektif bir şekilde değerlendirmek için filtreleme gerekmektedir.

Nasıl: Temizleme ve yedek alma işlemleri de kayıt altına alınmalıdır.

5.19 **Gerekli görüldüğü durumlarda, uygulama, yaygın kullanırlığı olan korelasyon yönetimi yazılımlarının kabul edebileceği formatlarda uygulama ve sistem denetim kayıtları tutabilmelidir.**

Etiket: İzleme, Denetim, Korelasyon

Neden: İzleme kayıtlarını anlamlandırmak ve efektif bir şekilde değerlendirmek için korelasyon araçlarından faydalanılabilir.

Nasıl: Bahsi geçen korelasyon yazılımlarına örnek olarak HP Arcsight, AlienVault OSSIM vs. verilebilir.

5.20 **Uygulama, yetkilendirilmiş kullanıcıların kayıt (log) arşivi saklama süresini ayarlayabilecekleri mekanizmalar sunmalıdır.**

Etiket: İzleme, Güvenlik Yönetimi

Neden: Kayıt arşivi saklama süresi değişik kanun ve mevzuatlara göre değişebilir.

Nasıl: Uygulama kayıtları belli bir periyoddan sonra (Örn. 48 saat) arşivlenmeli ve yeni bir kayıt dosyası açmalıdır. Veri tabanında tutulan kayıtlar için yeni kayıt dosyası açılması ya da süre zorunluluğu gerçekli olmayabilir. Kayıt arşiv dosyalarının (kayıtlar veri tabanında bile tutuluyor olsa) fiziksel olarak bağımsız başka bir alanda da tutulması önerilmektedir.

6. DİĞER GÜVENLİK ÖNLEMLERİ

6.1 Zararlı enjeksiyonlardan korunmak için uygulama veri girdisi denetimi yapmalıdır

Etiket: Sıkılaştırma, Saldırı Önleme

Neden: Enjeksiyon saldırılarında, saldırı vektörleri çok geniş ve çeşitlidir; SQL*, XSS, LDAP, Xpath*, XSLT*, XML*, OS* komutları ile gerçekleştirilebilir. Ancak savunma için alınacak birkaç temel önlem ciddi bir saldırının önüne geçebilir. Bu önlemler sadece enjeksiyon saldırılarının değil, *local file** veya *remote file inclusion** saldırılarının da önüne geçecektir. Bu önlemlerin en başında her türlü girdinin denetimi gelir;

Uygulama, birçok uçtan veri girdisi alabilir. Bunlar kullanıcılar, uygulamanın kullandığı alt yapı, veri tabanları veya 3. Parti diğer uygulamalar olabilir. Uygulama tüm girdilerini otomatik olarak denetleyerek, bu yolla uygulamaya zarar verilmesinin önüne geçmelidir. Zira saldırganlar, doğru girdi denetimi olmayan uygulamanın girdi uçlarını kullanarak çeşitli enjeksiyonlar, dosya sistemi saldırıları ve arabellek taşırma saldırıları gerçekleştirmek isteyeceklerdir. Denetimden geçip %100 güvenli olduğu doğrulanmayan hiçbir veriye güvenmemek gerekmektedir.

Nasıl: Uygulama, kullanıcılardan gelen tüm girdileri her katmanda denetlemelidir. Ancak, denetleme kodu, çalıştıran kodun fonksiyonuna göre uygulanmalıdır. Mesela, web / sunucu katmanı web ile alakalı sorunlar için denetlerken, kalıcı (persistence) katmanlar* SQL/HQL* enjeksiyonu gibi, dizin bulmalar LDAP enjeksiyonları gibi sorunları denetlemelidirler.

Uygulamada girdi denetiminin nasıl yapılacağına dair çözüm yolları aşağıdaki alt maddelerde sunulmuştur;

6.1.1 Uygulama tüm girdileri tek bir formata sokup işleme almalıdır. Bunun için yapılması gerekenler, önem sırasına göre,

1) Parametreli sorgular(paremetrized queries/Prepared statements),

2) Depolanmış prosedürler(stored procedures),

3) Liste üzerinden negatif veya pozitif girdi denetimi

şeklinde sıralanabilir. Aşağıda açıklamaları ve farklı yazılım dillerinde çözüm örneklerini bulabilirsiniz;

- **Parametreli Sorgular** bir SQL ifadesinin sorguya gitmeden yorumlanması(pre-compile) sonucu parametreler için yer tutucuların(placeholders) atanması ile asıl sorgunun sadece parametre değerleri aracılığıyla ayrıca yapılmasıdır. Sorguyu zararsız hale getirmesinin yanında diğer bir avantajı sorguyu hızlandırmasıdır. OWASP* web sitesinden alıntıladığımız değişik programlama dillerinden örnekleri şu şekildedir;

Dil - Kütüphane	Parametreli Sorgular
Java - Standard	String custname = request.getParameter("customerName"); String query = "SELECT account_balance FROM user_data WHERE user_name = ? "; **PreparedStatement pstmt = connection.prepareStatement(query);** **pstmt.setString(1, custname);** ResultSet results = pstmt.executeQuery( );
Java - Hibernate	Query safeHQLQuery = session.createQuery("from Inventory where productID=:productid"); safeHQLQuery.setParameter("productid", userSuppliedParameter);

.NET/C#	String query = "SELECT account_balance FROM user_data WHERE user_name = ?";
	try {
	OleDbCommand command = new OleDbCommand(query, connection);
	command.Parameters.Add(new OleDbParameter("customerName", CustomerName Name.Text));
	OleDbDataReader reader = command.ExecuteReader();
	// …
	} catch (OleDbException se) {
	// error handling
	}

ASP.NET	string sql = "SELECT * FROM Customers WHERE CustomerId = @CustomerId"; **SqlCommand command = new SqlCommand(sql);** **command.Parameters.Add(new SqlParameter("@CustomerId", System.Data.SqlDbType.Int));**

Dil - Kütüphane	Parametreli Sorgular
	command.Parameters["@CustomerId"].Value = 1;
Ruby - ActiveRecord	**# Create** Project.create!(:name => 'owasp') **# Read** Project.all(:conditions => "name = ?", name) Project.all(:conditions => { :name => name }) Project.where("name = :name", :name => name) **# Update** project.update_attributes(:name => 'owasp') **# Delete** Project.delete(:name => 'name')
Ruby	insert_new_user = db.prepare "INSERT INTO users (name, age, gender) VALUES (?, ? ,?)" insert_new_user.execute 'aizatto', '20', 'male'
PHP - PDO	$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)"); **$stmt->bindParam(':name', $name);** **$stmt->bindParam(':value', $value);**
Cold Fusion	<cfquery name = "getFirst" dataSource = "cfsnippets"> **SELECT * FROM #strDatabasePrefix#_courses WHERE intCourseID =** **<cfqueryparam value = #intCourseID# CFSQLType = "CF_SQL_INTEGER">** </cfquery>
Perl - DBI	my $sql = "INSERT INTO foo (bar, baz) VALUES (?, ?)"; **my $sth = $dbh->prepare($sql);** **$sth->execute($bar, $baz);**

Depolanmış Sorgular veri tabanı içinde yorumlanmış bir durumda, belli bir iş için atanmış ve birçok farklı program tarafından kullanabilen SQL cümlecikleridir. Bu depolanmış sorgular girdi denetimi yapacak şekilde ayarlanabilir. Örneğin;

VB .NET

```
Try
        Dim command As SqlCommand = new SqlCommand("sp_getAccountBalance", connection)
        command.CommandType = CommandType.StoredProcedure
        command.Parameters.Add(new SqlParameter("@CustomerName", CustomerName.Text))
        Dim reader As SqlDataReader = command.ExecuteReader()
        ' …
Catch se As SqlException
        ' error handling
End Try
```

JAVA:

```
String custname = request.getParameter("customerName"); // Bu girdi de doğrulanmalıdır.
try {
        CallableStatement cs = connection.prepareCall("{call sp_getAccountBalance(?)}");
        cs.setString(1, custname);
        ResultSet results = cs.executeQuery();
        // … sonuç
} catch (SQLException se) {
        // … loglama ve hata yönetimi
}
```

Negatif veya pozitif girdi denetimi potansiyel zararlı girdi değerlerinin bir liste halinde yazılıp hiç değerlendirmeye alınmaması veya sadece kabul edilecek girdi değerlerini belirleyip, bunların dışındaki tüm verileri reddederek gerçekleştirilir. Pozitif girdi denetimi çok daha verimli bir metottur. Eğer beyaz veya siyah listeyle girdi denetimi yapılıyorsa, incelenmesi gereken hususlar şunlardır:

- Veri tipleri(örn. String, integer, real…)
- İzin verilen karakter setleri
- Minimum ve maksimum uzunluk
- Çift kayıta(duplicates) izin verilip verilmeyeceği
- Rakamsal alan
- Özel şablonlar(örn. RegEx)

Negatif girdi denetimi örneği:

```
<?php
$searchparam = $_POST['searchparam'];
$key = array('<script>', '</script>');
$replace = array(' ', ' ');
$searchparam = str_replace ($key, $replace, $searchparam);

echo searchparam;
?>
```

Pozitif girdi denetim örneği:

```
<?php
$searchparam = $_POST[searchparam];
if(!preg_match('[-_ 0-9A-Za-z] ', $searchparam))
{
header( 'Location: error.php' );
}
else{ echo searchparam;
?>
```

6.1.2 Java geliştirirken ESAPI, .Net geliştirirken .NET Framework gibi üst ana çatılar kullanılıyorsa, girdi denetimi önceden tanımlanmış fonksiyonlar aracılığıyla çok rahat sağlanabilir.

Örneğin; ESAPI'de encodeForSQL metodu kullanılabilir;

```
Codec codec = new OracleCodec();
String input = req.getParameter("girdi");
ESAPI.encoder().encodeForSQL(codec, girdi);
```

.NET Framework için: "validateRequest" bayrağı "true" olarak işaretlendiğinde, girdi denetimi otomatik olarak yapılacaktır.

6.1.3 Uygulama, tüm başlık, çerez, sorgu cümleciği, form alanı ve gizli alanlar için denetim(validation) uygulamalıdır.

6.1.4 Uygulama, kullanıcıdan gelen girdileri XSS önlemi olarak kodlamalarıdır(encode).

Tüm kullanıcı girdileri dışa aktarılmadan önce özel karakterlerin kodlanması(escaping) gerekmektedir. Örneğin ;

<script> -> <script>

Bunun yanında http cevaplarında XSS'e karşı geliştirilmiş bir W3C standardı olan CSP (Content Security Policy) header bilgileri kullanılmalıdır. Tüm tarayıcılar, CSP kullanılan http cevaplarını tespit edince, javascript kodlarını daha fazla kurallara bağlı olarak çalıştıracaklardır.

6.1.5 Uygulama, güvenli(type-safe) SQL parametreleri kullanmalıdır. Girdiler çalıştırılacak bir kod olarak değil, harf dizini olarak değerlendirilmelidirler.

6.1.6 Uygulama, özel anlamlar ifade eden kaçış karakterleri ekleyerek kodu temizleyecek(sanitize) filtreleme rutinleri kullanmalıdır.

Çıktı kodlaması için var olan kütüphaneleri kullanmak gerekmektedir. Bunlar;

- .NET Anti-XSS kütüphanesi (AntiXSS.getSafeHTML)
- ASP.Net ValidateRequest (sınırlı koruma)
- ESAPI Encoder (encodeFor metotları)
- OWASP Java Encoder
- OWASP HTML Sanitizer
- JSF : <h:outputText value="#{user.name}" escape="true"/>
- Spring MVC : <form:input path="someFormField" htmlEscape="true" />
- PHP htmlpurifier: http://htmlpurifier.org/
- Python Bleach: https://pypi.python.org/pypi/bleach
- JavaScript: https://github.com/asutherland/bleach.js/blob/master/lib/bleach.js

6.1.7 Enjeksiyonlardan kaçınmak için API'ler kullanılmalıdır. Örneğin Nesne ile İlişkisel Eşleme(Object to Relational Mapping)(ORM) kütüphaneleri, düşük düzeydeki veri tabanı işlemleri ve veri tabanından gelen uyumsuz veya saldırgan veri tipleri ile uğraşmadan programlama dilinin sınıfları ile veri tabanı üzerinde işlem yapabilmek için kullanılabilir. Ancak ORM yanlış kullanıldığında yorumlayıcı(interpreter) enjeksiyon saldırılarına yol açabileceği dikkate alınmalıdır.

/* Positional parameter in HQL */

Query hqlQuery = session.createQuery("from Orders as orders where orders.id = ?");

List results = hqlQuery.setString(0, "123-ADB-567-QTWYTFDL").list();

/* named parameter list in HQL */

List items = new ArrayList();

items.add("book"); items.add("clock"); items.add("ink");

List results = session.createQuery("from Cart as cart where cart.item in (:itemList)").setParameterList("itemList", items).list();

/* JavaBean in HQL */

Query hqlQuery = session.createQuery("from Books as books where book.name = :name and book.author = :author");

List results = hqlQuery.setProperties(javaBean).list(); //assumes javaBean has getName() & getAuthor() methods.

/* Native-SQL */

Query sqlQuery = session.createSQLQuery("Select * from Books where author = ?");

List results = sqlQuery.setString(0, "Charles Dickens").list();

********** Yanlış Kullanım **********

List results = session.createQuery("from Orders as orders where orders.id = " + currentOrder.getId()).list();

List results = session.createSQLQuery("Select * from Books where author = " + book.getAuthor()).list();

6.1.8 LDAP enjeksiyonlarından kaçınmak için, gerekli kullanıcı isimleri dışındaki her şeyi elemine etmek için pozitif doğrulama yapmak gereklidir.

```
// Kullanıcı adı girdisinde özel LDAP karakterleri mevcut ise, istisna atıp
dur.

if ( containsLDAPspecials(getParameter("username")) ) {

        throw new javax.naming.AuthenticationException();

}



String principal = "cn=" + getParameter("username") + ", ou=Users,
o=example";

String password = getParameter("password");



env.put(Context.SECURITY_AUTHENTICATION, "simple");

env.put(Context.SECURITY_PRINCIPAL, principal);

env.put(Context.SECURITY_CREDENTIALS, password);



// İlk konteksti oluştur.

DirContext ctx = new InitialDirContext(env);
```

6.1.9 Uygulamada, kök dizini dışında saklanan dosya ve dizinlere yetkisiz ulaşımı amaçlayan Yol Gezinimi (path traversal) saldırılarını engellemek için sistem çağrılarını kullanıcı girdisiyle yapmaktan kaçınmak gerekir. Ayrıca adresleme yaparken, dosya isimlerini kullanmaktan ziyade indexlerin kullanılması gerekmektedir.

6.1.10 Ara Bellek Taşması'nı (Buffer Overflow) engellemek için, eğer aksi gerekli değilse Assembly, C ve C++ yerine, bellek yönetimi ve bellek taşması kontrolü yapan C# ve Java gibi üst seviyeli dillerle programlama yapmak tercih edilmelidir. Bu mümkün değilse zafiyete neden olmayacak güvenli fonksiyonlar kullanılarak önlenmelidirler. Örneğin;

Güvensiz Fonksiyonlar	**Muadili Güvenli Fonksiyonlar**
Strcat	Strlcat
Strcpy	Stlcpy
Stmcat	Strlcat
Stmcpy	Strlcpy
Sprintf	Snprintf
Vsprintf	Vsnprintf
Gets	Fgets

Tablo 1: Ara bellek taşmasına sebep olabilecek fonksiyonlar ve güvenli muadilleri

Bunlara ilaveten, yukarıda ayrıntılı açıklaması yapılan girdi denetimi burada da uygulanmalıdır. Ayrıca kod yazılırken değişken büyüklüğünü her seferinde sayıyla girmek yerine bir kereliğine bir değişkene ya da sabite atayıp sonra hep o kullanılmalı veya değişkene yazılabilecek karakter miktarı "sizeof" fonksiyonuyla hesaplanmalıdır. Kod derlenirken derleyicilerin sağladığı arabellek taşması tespit mekanizmaları aktif hale getirilmelidir. ASLR(Address Space Layout Randomization)* ve DEP(Data Execution Protection)* gibi özellikler kullanılmalıdır.

Son olarak; Arabellek taşması güvenliği sağlayan kütüphane ve üst anaçatılar(framework) kullanılmalıdır.

6.1.11 Tüm denetimler hem istemci, hem de sunucu tarafında yapılmalıdır.

Zira istemci tarafında yapılan denetimler, saldırganların Proxy yazılımlarından faydalanarak istemci ile sunucu arasında girmeleri sonucu kolay bir şekilde geçersiz kılınabilmektedir. Bu denetimler ASP .NET düzenli ifadeleri(regular expressions)* şu şekilde yapılabilir;

İstemci tarafı:

```
<asp:RegularExpressionValidator runat=”server” ID=”searchValidator”
ControlToValidate=”TxTSearch” ErrorMessage=”Special characters are not allowed”
ValidationExpression=”/^[-_ 0-9a-z]|\.|[-_ 0-9a-z]$/i” />
```

Bu regular expression validasyonu ile sadece alfanümerik karakterler ile arama yapılması, aksi takdirde “Special characters are not allowed” hata mesajının verilmesi sağlanmıştır.

Sunucu tarafı:

aspx dosyasının c# dosyasının başına,

```
using System.Text.RegularExpressions;
```

yazılarak RegularExpression class’ı import edilmelidir. Ardından dosya içerisinde gerekli denetimin yapılacağı yerde aşağıdaki kod yazılırak denetim sağlanabilir.

```
Regex regex = new Regex(“^[a-zA-Z0-9]*$”);
if (regex.Match(queryString).Success)
{
//Do something!
}
else{
//Do something!
}
```

6.2 Siteler Arası İstek Sahteciliği(Cross Site Request Forgery)(CSRF) önlenmelidir

Etiket: Sıkılaştırma, Saldırı Önleme

Neden: CSRF, zararlı bir web sitesi, email veya programın, tarayıcının, kullanıcının hali hazırda kimlik doğrulaması yapmış olduğu güvenilir bir site üzerinde istem dışı hareketler yapmasına zemin

hazırlayan saldırı çeşididir. Bu saldırının başarısı, kimliği doğrulanmış kullanıcının sistemdeki yetkileriyle doğru orantılıdır. Otomatik açıklık tarayıcıları ile tespiti zordur.

CSRF saldırısının altında yatan en temel problem, kullanıcı tarafından gönderilen talepler ile tarayıcı tarafından otomatik olarak gönderilen taleplerin sunucu tarafından birbirinden ayrılamaması ve sunucunun bütün talepleri kullanıcıdan geliyor gibi işleme almasıdır.

Nasıl: Bu problemin çözümü olarak;

- Sunucunun, kullanıcıya gönderdiği HTTP cevabının içerisinde kritik işlemlerin yapıldığı form alanları ve/veya bağlantılar içerisine rastgele üretilmiş tanımlayıcılar koyularak istemci tarafından gelen tüm taleplerde bu rastgele tanımlayıcı ifadelerin kontrol edilmesi sağlanabilir. (Aşağıdaki maddelerden uygun olan seçilerek uygulanabilir)
    - Adres içerisinde rasgele tanımlayıcılar: /servlet/f81d4fae-7dec/ManageUser
    - Parametre olaran verilen rastgele tanımlayıcılar: /servlet/ManageUser?csrfid= f81d4fae-7dec
    - Form objesi içerisinde bulunan rastgele değerli gizli girdi alanları:

```
<form action="/transfer.do" method="post">
        <input type="hidden" name="CSRFToken" value=" f81d4fae-7dec”>
        <input type="text" name="Username">
        <input type="password" name="Password">
</form>
```

- Referer başlığı kontrol edilerek özellikle HTTPS kullanılmayan uygulamalarda CSRF kontrolü sağlanabilir. Talebin geldiği web sitesinin belirtildiği Referer başlığı sayesinde bir talebin kullanıcıdan gelip gelmediği kontrol edilebilir.
- OWASP CSRF Guard çatısı kullanılarak hızlı ve güvenli bir çözüm üretilebilir.

Referans:

- http://www.bilgiguvenligi.gov.tr/web-guvenligi/webdeki-buyuk-tehlike-csrf.html
- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

6.3 Uygulamanın kullandığı 3. Parti kütüphanelerin, COTS ürünlerin vs. açıklık içermediği bilinen en güncel sürümlerinin kullanıldığı garanti edilmelidir.

Etiket: Sıkılaştırma

Neden: Modern uygulamalar birçok farklı modülden oluşur ve geliştirilirken farklı kütüphanelerden faydalanılır. Dışarıdan uygulamaya monte edilmiş bu tarz kütüphane veya modüllerin güvenlik açıklıkları saldırganlar tarafından istismar edilebilir.

Nasıl: Mümkün olduğu kadar açık kaynaklı kütüphane veya teknoloji kullanılması tavsiye edilir. Kullanılan tüm 3. Parti ürünler, ister açık kaynaklı, ister COTS olsun, periyodik olarak güncellenmelidirler.

6.4 Uygulama, sunucu hatalarının açığa çıkmasını engellemek için kullanıcı için özel hata mesajları hazırlamalıdır.

Etiket: Sıkılaştırma, Hata Yönetimi

Neden: Saldırganlar hata mesajlarından edindikleri bilgileri uygulamayı zaafa uğratmak için kullanabilirler.

Nasıl: Kullanıcılara sunucu hatası mesajı verilirken, uygulama hakkında mümkün olduğu kadar az bilgi vermek gerekmektedir. Sadece hatanın ne olduğu ve hangi sayfada çıktığı bilgisi yeterli olabilir.

6.5 Uygulamada geliştirilirken göz önünde bulundurulmamış ya da kod içinde yakalanmamış hata durumları oluşabileceğinden, yakalanmamış hatalar nedeniyle bilgi ifşası ya da benzeri zafiyetlerin önüne geçilmesi için gerekli önlemler alınmalıdır.

Etiket: Sıkılaştırma, Hata Yönetimi

Neden: Saldırganlar, uygulamaya geliştirici tarafından ön görülemeyen hatalar yaptırtmaya ve bu hataları istismar etmeye çalışırlar.

Nasıl: Örneğin .NET ortamında yazılmış bir uygulama için “web.config” dosyasına aşağıdaki satır eklenebilir. Beklenmedik bir hata oluşması durumunda uzaktan bağlanan kullanıcılara kısıtlı hata mesajı gösterilecektir.

```
<compilation defaultLanguage="c#" debug="false" /> <customErrors mode="RemoteOnly" />
```

6.6 Uygulama, işletim sistemi veya veri tabanı seviyesindeki hatalar ile karşılaştığında, kullanıcıya hata tipini de içeren bir mesaj iletmelidir.

Etiket: Hata Yönetimi

Neden: Uygulamanın dışında, fakat uygulamanın bağlı olduğu birimlerde ortaya çıkabilecek hatalar, uygulamanın güvenliğini de tehdit edebilir.

Nasıl: Minimum bilgi prensibi* doğrultusunda kullanıcıya sadece hata kodu ile hata ile nerede karşılaşıldığı bilgisini vermek yeterli olacaktır.

Diğer Önlemler

6.7 Uygulama, kimlik doğrulama aşamasından sonra gelen kontrollü taramayı(forced browsing) engellemelidir.

Etiket: Sıkılaştırma, Saldırı Önleme

Neden: Kontrollü tarama, uygulama üzerinde link tanımı olmayan fakat aktif sayfaların saldırgan tarafından bulunması ve istismar edilmesidir.

Nasıl: Saldırgan, kontrollü tarama yapmak için kaba kuvvet saldırıları ile domain dizini altındaki geçici dosya, eski yedekler ve ayar dosyalarını arayacaktır. Bunun önüne geçmek için link vermediğiniz sayfalarınızın isimlerini tahmin edilemeyecek isimlerle değiştirin. Örneğin http://siteniz.com/uygulamanız/admin.jsp yerine http://siteniz.com/uygulamanız/uyg-yonet.jsp gibi bir adresleme yapılabilir.

6.8 Uygulama, kendi oluşturduğu dosyaların izinlerinin bozulması durumunda, yetkileri eski haline getirmeyi mümkün kılan mekanizmalar sunmalıdır.

Etiket: Yetkilendirme, Güvenlik Yönetimi

Neden: Uygulama, kullanım süresi boyunca kendisi için dosyalar ve dizinler oluşturacaktır. Farklı müdahaleler ile bozulan bu dosya izinleri, uygulamayı saldırganların istismarına açık hale getirebilir.

Nasıl: Yönetici paneline konulacak, “dosya/dizin izinlerini varsayılana çek” butonu ile çözüm sağlanabilir.

6.9 Uygulama istemci taraflı veri saklama suretiyle oluşabilecek muhtemel veri gizliliği ihlallerinin önüne geçmelidir.

Etiket: Sıkılaştırma, Gizlilik

Neden: İstemci tarafında kaydedilen her tür veri tehdit altındadır.

Nasıl: Rich client(silverlight, flash, applet vs.) içinde hassas bilgiler bulunabileceği için istemci tarafına gönderilen herhangi bir veride kullanıcının ihtiyacı olmayan bilgi bulunmamalıdır.(örneğin jar dosyaları içinde ip adresleri, dosya yolları, kullanıcı bilgileri vs. bulunmamalıdır.)

GÜVENLİK YÖNETİMİ

6.10 Uygulamaya en az bir adet sistem yöneticisi* ve/veya bir adet güvenlik yöneticisi* hesabı eklenmesi önerilir.

Etiket: Yetkilendirme, Güvenlik Yönetimi

Neden: Uygulama güvenlik ayarlarının yapılması, kullanıcı hesabı oluşturma ve yetkilendirmesinin yapılması için en az bir tane yetkili hesap gerekmektedir.

Nasıl: Uygulama geliştirilirken yönetici hesaplarının eklenmesi kabiliyetleri ile beraber tasarlanmalıdır.

6.11 Uygulamanın içinde detaylı bir yapılandırma ve ayar kontrol sistemi üretilmelidir.

Etiket: Denetim, Güvenlik Yönetimi

Neden: Güvenlik ayarlarının düzgün yapılamadığı uygulamalar saldırılara karşı zayıftırlar.

Nasıl: Tüm sistem bileşenleri için güvenlik fonksiyonları merkezi olarak yapılandırılmalıdır. Ara yüzden yapılandırma gerçekleştirilemiyorsa, en azından ara yüzde bilgi sunulması ve link verilmesi gerekir.

6.12 Uygulama, düzenli bir şekilde üzerine güvenliği onaylanmamış donanım veya yazılım eklemesi olup olmadığını kontrol etmeli, böyle bir şey söz konusu ise güvenlik veya sistem yöneticilerini uyarmalıdır.

Etiket: Bütünlük, Güvenlik Yönetimi

Neden: Uygulamanın içerden veya dışarıdan müdahale edilmemiş bir durumda çalıştığını garanti edebilmek, güvenlik adına önemlidir.

Nasıl: Müşteri ileri seviyede uygulama bütünlüğü talep ederse, uygulamanın kullandığı alt bileşenler ve uygulamanın kullandığı tüm yapılandırma dosyaları ve 3. Parti uygulamaların hepsinin tercih edilen bir metotla bütünlüğünün sağlanması gerekir. Bunun için örneğin checksum veya dijital imza kullanılabilir.

6.13 Müşterinin talep etmesi durumunda, uygulama veya kritik uygulama bileşenlerinden herhangi biri için güncelleme gerektiğinde, sistem veya güvenlik yöneticilerini otomatik olarak uyarılmalıdır.

Etiket: Güvenlik Yönetimi

Neden: Güncellenmemiş uygulama birimleri güvenlik zafiyetine sebep olurlar.

Nasıl: Uygulama, kullandığı bütün COTS ürünlerin güncelliklerini periyodik olarak kontrol etmeli ve gerektiğinde kullanıcı uyarılmalı ve otomatik güncelleme seçeneği sunulmalı.

6.14 Uygulama, kendisi ile bağdaştırılacak yeni bileşenlerin yüklenmesini sadece yetkili kullanıcıların yapabileceği şekilde kısıtlamalıdır.

Etiket: Güvenlik Yönetimi, Yetkilendirme

Neden: Yetkisiz yapılan eklentiler büyük güvenlik zafiyetlerine sebep olabilir.

Nasıl: Madde 3'teki yetkilendirme adımları uygulanmalıdır.

6.15 Güvenlik ile alakalı olaylar, bu olayların iletişimi ve atılan düzeltme adımları, uygulama tarafından kayıt altına alınmalıdır.

Etiket: İzleme, Denetim, Güvenlik Yönetimi

Neden: Uygulamada izlenmesi gereken en önemli hareketler güvenlik ayarlarında yapılacak değişikliklerdir. Aynı zamanda, bir saldırgan uygulamaya sızarsa, yaptığı değişikliklerin kayıt altında olması uygulamayı eski haline getirmek için önemlidir.

Nasıl: Madde 5'teki izleme ve denetim adımları, güvenlik ile alakalı olayları da kapsamalıdır. Bu olaylar ile alakalı kayıt altına alınması gerekenler aşağıdaki alt maddelerde listelenmişlerdir;

6.15.1 Olay türü

6.15.2 Ne zaman oluştu

6.15.3 Kime bilgi verildi

6.15.4 Olayın etkileri

6.15.5 Düzeltme Adımları

7. Sözlük

- **Ara Vekil(Proxy)**

Bilgisayar ağlarında, bir vekil sunucu diğer sunuculardan kaynakları isteyen istemcilerin talepleri için bir aracı olarak davranan sunucudur. Bir istemci vekil sunucuya bağlanır, bazı servisler ister, örneğin bir dosya, bağlantı, ağ sayfası veya farklı bir sunucudan uygun diğer kaynaklar gibi ve vekil sunucusu, kolaylaştırmak ve karmaşıklığını kontrol etmek için bir yol olarak talebi değerlendirir. Bugün birçok vekil, ağ vekilleridir.

- **ASLR(Adress Space Layout Randomization)**

Çalıştırılabilir dosyaların, her çalışma esnasında sanal adres alanında oluşturulan adres değerlerinin rasgele olarak belirlendiği bir mekanizmadır. Bu yapı, saldırganların uygulamanın sanal adres alanındaki yerlerini tespit edebilmesini ve zararlı kod çalıştırmasını oldukça güçleştirmektedir.

- **Bağlantı Havuzu(Connection Pooling)**

Connection Pooling, veri tabanında bağlantıların bir havuza atılarak buradan kullanılmalarını sağlayarak, çok sayıda kullanıcının bağlı olduğu veri tabanlı uygulamalarda, aynı özelliklere sahip bağlantı bilgilerinin defalarca oluşturulmasının önüne geçen ve bu sayede var olan açık bağlantıların kullanılabilmesini sağlayan mekanizmadır.

- **Bütünlük**

Veri bütünlüğünde amaç, veriyi olması gerektiği şekilde tutmak ve korumaktır. Bilginin bozulmasını, değiştirilmesini, yeni veriler eklenmesini, bir kısmının veya tamamının silinmesini engellemeyi hedefler. Bu durumda veri, haberleşme sırasında izlediği yollarda değiştirilmemiş, araya yeni veriler eklenmemiş, belli bir kısmı ya da tamamı tekrar edilmemiş ve sırası değiştirilmemiş şekilde alıcısına ulaşır.

- **CAPTCHA**

Completely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part, bilgisayar ve insanın ayırt edilmesi için kullanılan teknolojidir.

- **COTS(Commercial Off-the-shelf)**

Devlet veya bir firmanın kendisi veya ürünleri için satın aldığı, ticari olarak alış-verişi mümkün olan ürün.

- **Çerez(Cookie)**

Web uygulamaları kimlik doğrulaması adımından başarı ile geçen kullanıcılar oturumlarını saklamak için Session ID/Cookie (çerez) bilgisini kullanırlar. Yaratılan bir çerezin içeriğinde kullanıcıya ait oturum kimliği de bulunur. Bu oturum kimlikleri kullanıcı başarılı bir kimlik doğrulama adımını geçtikten sonra özel olarak hazırlanmış tahmin edilmesi zor Session ID'lerinden oluşturulurlar ve hem sunucu tarafında hem de kullanıcı tarafında saklanırlar.

- **DEP(Data Execution Prevention)**

Modern işletim sistemlerine dahil edilmiş bir güvenlik mekanizmasıdır; program, servis veya sürücülerin çalıştırılabilir yönergeleri(exevutable instructions) dışında bir veri içeren hafıza bölümüne bloke eder.

- **Dijital İmza**

Başka bir elektronik veriye eklenen veya elektronik veriyle mantıksal bağlantısı bulunan ve kimlik doğrulama amacıyla kullanılan elektronik veridir. Dijital imza, imza sahibinin kimliğini imzalanan veriyle ilişkilendirir ve imzalanan verinin değiştirilmediğini ispat eder.

- **DOS(Denial of Service) Saldırıları**

İnternet üzerinden çok sayıda istemci bilgisayar kullanılarak yapılan bir saldırı çeşididir. Çoğunlukla virüs bulaştırılarak zombi haline getirilen bilgisayarların, bir sunucu bilgisayara eş zamanlı ve mümkün olduğunca çok sayıda istek göndermesi, sunucunun kapasitesinin aşılması sonucunda da hizmet veremez hale getirilmesi ilkesine dayanır.

- **Düzenli İfadeler(Regular Expressions)**

Ele alınan metindeki kimi katarların kısa yoldan ve esnek bir biçimde belirlenmesini sağlar. Bu katarlar belli karakterler, kelimeler veya karakter örüntüleri olabilir. Düzenlemeli ifadeler, bir biçimsel dil kullanarak yazılır ve bir düzenlemeli ifade işleyici tarafından yorumlanır. Bir düzenlemeli ifade işleyici, ya ayrıştırıcı üreteci olarak hizmet eden ya da metni inceleyip verilen tarife uygun kısımlarını belirleyen bir programdır.

- **Erişilebilirlik:**

Bilginin her an ulaşılabilir ve kullanılabilir olmasını amaçlayan prensiptir. Bilişim sistemlerinin kendilerinden beklenen işi sürekli bir şekilde tam ve eksiksiz olarak yapmasını amaçlamaktadır. Süreklilik hizmeti, bilişim sistemlerini, kurum içinden ve dışından gelebilecek başarım düşürücü

tehditlere karşı korumayı hedefler. Süreklilik hizmeti sayesinde, kullanıcılar, erişim yetkileri dâhilinde olan verilere, veri tazeliğini yitirmeden, zamanında ve güvenilir bir şekilde ulaşabilirler.

- **Erişim Kontrol Matrisi**

Erişim kontrolü olan uygulamalarda, kullanıcıların bir sayfayı görüp göremeyeceğini veya kullanıcıların bir aksiyon gerçekleştirmeden önce buna hakkı olup olmadığına dair kontrol listesidir.

- **ERP:**

Kaynak planlaması yazılımı.

- **Get/Post Değişkenleri:**

Kaynak isteme ve gönderme için kullanılan http talepleridir.

- **Gizlilik**

Bilginin yetkisiz kişilerin eline geçmesini engellemeyi amaçlamaktadır. Bilgi hem bilgisayar sistemlerinde, hem saklama ortamlarında, hem de ağ üzerinde gönderici ve alıcı arasında taşınırken yetkisiz erişimlerden korunmalıdır. Saldırgan bir yapılandırma veya yazılım hatasını istismar ederek yahut Sosyal Mühendislik teknikleri ile yetkili insanların hatalarını istismar ederek bilgilere izinsiz olarak erişebilir.

- **Gökkuşağı Saldırısı**

Temelde bir sözlük saldırısıdır. Farkı, sözlük içinden direkt olarak parolaların değil, parola özetlerinin teker teker denenmesidir.

- **Güvenlik İsterleri Dokümanı(GİD):**

Güvenli Yazılım Yaşam Döngüsünün en başında, müşteri ve yüklenicinin beraber oluşturdukları, uygulamanın güvenliği için belirlenen isterler listesidir.

- **HTTP Başlıkları(Http Headers)**

http protokolündeki istek ve yanıt mesajların başlık bölümlerine denir. http protokolü alış-verişinde işleme konulacak parametreleri belirlerler.

- **IPSec:**

İnternet ortamında verinin gizliliğinin ve bütünlüğünün korunabilmesi için IP seviyesinde güvenlik gereksinimi IPv4 ortamında bu IPSec standardı ile sağlanabilmektedir. Fakat bu standardın IPv4 uygulamalarında bulunması zorunlu değildir.

- **İş Nesnesi(Business Object)**

N-katmanlı nesne yönelimli yazılımların iş(business) katmanında bulunan aktör.

- **İzleme ve Denetim**

Bir sorun ile karşılaşıldığında sorunun tespitinin sağlanabilmesi için kullanılan sistemdir. Sistemde bulunan kullanıcıların yaptıkları bütün işlemler ve erişim saatleri kayıt altına alınır. Bir problem çıktığında ise kullanıcı aktivitelerinin tutulduğu bu kayıtlardan sorun anlaşılmaya ve çözülmeye çalışılır.

- **Kaba Kuvvet(Brute Force) Saldırısı**

Belli bir hesaba bir veya birden çok bilginin/parolanın denenmesiyle o hesabı elde etmeye çalışma yönteminin adıdır.

- **Kalıcı Katmanlar(Persistent Layers)**

Kalıcı katmanlar, bir işlemin oluşturduğu durumun(state), işlem sona erdikten sonra da devam ettiği yazılım katmanıdır. Eğer bu kalıcılık sağlanamamış olsaydı, hedeflenen durumlar işlemle beraber sadece RAM'de var olacak, RAM gücünü kaybettiğinde kayıp olacaklardır.

- **Kimlik Doğrulama:**

Kullanıcının sisteme bağlanabilmesi için ilk başta yapılması gereken işlemdir. Sistem kullanımı sırasında cihaz veya kullanıcının kimliğinin doğrulanmasıdır. Bu işlem ile kullanıcının sahip olduğu kullanıcı adının sistemde kayıtlı olup olmadığı kontrol edilir. Daha sonra kullanıcıya verilen parola da kontrol edilerek doğrulama işlemi yapılır. Doğrulama sağlanırsa kullanıcıya sisteme giriş izni verilir. Bilgisayar ağları ve bilgisayar sistemleri dışında fiziksel sistemler için de çok önemlidir ve bu yüzden akıllı kartlar veya biometrik teknolojilere dayalı kimlik sınama sistemleri kullanılamaya başlanmıştır.

- **Kripto Ürünü**

Kriptoprafik şifreleme işlemleri yapma kapasitesi olan araçlar.

- **LDAP(Lightweight Directory Access Protocol)**

LDAP (Türkçe: Basit Dizin Erişim Protokolü) TCP/IP üzerinde çalışan dizin servislerini sorgulama ve değiştirme amacıyla kullanılan uygulama katmanı protokolüdür. Bu protokol, OpenLDAP, Sun Directory Server, Microsoft Active Directory gibi dizin sunucuları tarafından kullanılmaktadır.

- **Local File Inclusion**

Genellikle uygulamaların dosya aktarımında kullandıkları formlarda görülen bir problemdir. Eğer dosya tipi ve içeriği kontrolü tam olarak yapılmıyorsa, zararlı kod parçası bulunduran dosya sistem üzerine

yüklenir ve açıklık bulunduran girdi noktası kullanılarak yüklenen zararlı yazılım çağrılıp çalıştırılarak sisteme sızılır.

- **Mantıksal Silme(Soft Delete)**

Mantıksal silme ile veriler fiziksel olarak sistemden kazınmak yerine başka bir bölüme alınır ve varlıkları devam eder.

- **Mantıksal Veri(Logical Data)**

Verinin, veri tabanlarındaki tablolarda sanal temsillerine denir.

- **MITM Saldırıları(Man-in-the-middle)**

Bu saldırı türü ile güvensiz ortamlarda (şifrelenmemiş) yapılan bağlantılarda tüm trafik kontrol edilebilip içeriği değiştirilerek manipüle edilebilir. Yani açmak istenilen sayfadan farklı bir sayfa açılabilir, gönderilen e-postanın içeriği değiştirilip alıcıya farklı bir mesaj gönderilebilir. Fakat güvenli bağlantı(ssl) kullanılıyorsa, bu saldırının gerçekleşmesi güçleşir.

- **Nesne(Objects)**

Nesne Yönelimli Programlamada, kendi içerisinde veri işleyebilen ve diğer nesneler ile çift yönlü veri alışverişinde bulunabilen etkileşim birimleridir.

- **OS(Operating System)**

İşletim Sistemi.

- **Oturum Sabitleme(Session Fixation)**

Bu saldırıda ise saldırgan bildiği bir oturum çerezinin kullanıcı tarafından oturum açma işlemi gerçekleşmeden kullanılmasını sağlar ve böylece kullanıcının saldırgan tarafından atanmış çerez ile oturum açması sonucu oturumu ele geçirir veya kendi oturumundan devam etmesini sağladığı kullanıcının bilgi gizliliğini bozabilir. Bu duruma örnek olarak, kullanıcının kendi oturumunun değiştiğini fark etmemesi ve kendi (kredi kartı, özlük, vs.) bilgilerini saldırgana ait oturuma girerek kaydetmesi gösterilebilir.

- **Referrer Başlığı:**

Get talebi ile gönderilen, sayfaya nereden ulaşıldığını işaretleyen opsiyonel bir değişkendir.

- **Remote File Inclusion**

Zararlı kod barındıran dosyaların uygulamaya uzak sunucu yolu belirterek eklenmesi sonucu ortaya çıkan zafiyet çeşididir.

- **RIPEMD**

Kriptografik özet fonksiyonları ailesidir.

- **SHA**

Özetleme fonksiyonlarından olan SHA herhangi bir uzunluktaki bir metinin sabit uzunluktaki özetini oluşturur.

- **Sharepoint:**

Cihazlardaki bilgileri depolamak, organize etmek, paylaşmak ve erişmek için güvenli bir yer olarak kullanabilen Microsoft sunucu çözümüdür.

- **Simple Object Access Protocol (SOAP)**

Simple Object Access Protocol(SOAP), servisler arasında yapılandırılmış bilgi alışverişini sağlayan XML-tabanlı bir iskelet sağlar. Bu bilgi, Header(Başlık) ve Body(Gövde) şeklinde biçimlenir, teorik olarak birçok iletişim protokolü üzerinden taşınabilir, fakat sadece HTTP-bağı resmi olarak tanımlanmıştır ve bugün kullanımdadır. SOAP Remote Procedure Call(RPC, Uzak Prosedür Çağrısı) stilinde etkileşim sağlar ve uzak fonksiyon çağrıları ve Döküman-stili iletişim gibi ve mesaj içeriği sadece WSDL'deki XML Şema tanımına göre hazırlanır. Çağrı sonuçları isteğe bağlı olarak cevap mesajı ile birlikte döndürülür veya hata mesajı döndürülebilir ki bu geleneksel programlama dillerindekine kabaca benzerdir.

- **Sözlük Saldırısı**

Kripto analiz ve bilgi güvenliğinde bir şifreyi çözme veya kimlik doğrulama mekanizmasını baypas etmek için kullanılır. Deşifre için gereken anahtar, sözlük içindeki kelimelere benzeyen milyonlarca olası anahtarın teker teker denenmesiyle bulunmaya çalışılır.

- **SSL(HTTPS)**

SSL; istemci ile sunucu arasındaki trafiği şifreleyerek güvenli haberleşmeyi sağlayan bir güvenlik protokolüdür. Şifrelenmiş oturumu kurmak için el sıkışma(SSL handshake) yöntemi kullanır ve bu işlem sırasında sertifika doğrulaması, kullanılacak şifreleme algoritmaları ve desteklenen versiyon bilgileri vs. belirlenir. Bu protokolün çalışması sırasında uygulanan sertifika doğrulama işlemi, sertifikayı web

sitesinin kimliği şeklinde ifade edersek istemci ile sunucu arasında yapılacak olan haberleşmenin gerçekten doğru kişiler arasında mı olduğunu ispatlamak için uygulanır.

- **SQL**

SQL, (İngilizce "Structured Query Language", Türkçe: Yapılandırılmış Sorgu Dili) verileri yönetmek ve tasarlamak için kullanılan bir verit abanı yönetim sistemidir.

- **Tek Giriş(Single Sign-On)(SSO)**

Kimlik doğrulama işleminin, ekosistem dahilindeki bütün uygulamalar için merkezileştirilmesi olarak tanımlanabilir.

- **Uygulama veri dağıtım servisleri(Data Distribution Services) (DDS)**

OMG grubu tarafından belirlenen publish/subscribe temelli, gerçek zamanlı dağıtık sistemlerin geliştirilmesi için önerilen, makineler arası middleware standardı.

- **XML**

Extensible Markup Language (Genişletilebilir İşaretleme Dili, kısaca XML), hem insanlar hem bilgi işlem sistemleri tarafından kolayca okunabilecek dokümanlar oluşturmaya yarayan, W3C tarafından tanımlanmış bir standarttır. Bu özelliği ile veri saklamanın yanında farklı sistemler arasında veri alışverişi yapmaya yarayan bir ara format görevi de görür. SGML'in basitleştirilmiş bir alt kümesidir.

- **XPATH**

XPath bir XML dokümanındaki bilgiyi bulmak için kullanılan bir dildir. XPath bir XML dokümanı içindeki elemanları ve onlara ait özellikleri incelemeye yarar.

- **XSLT**

XSLT(Extensible Stylesheet Language Transformations, Genişletilebilir Biçimlendirme Dili Dönüşümleri), XML tabanlı, XML dokümanlarını dönüştürmek için kullanılan bir dildir. Orijinal dokümanı değiştirmeden, yeni bir doküman oluşturmaya olanak sağlar.

- **XSS (Cross-site Scripting)**

Kötü niyetli kullanıcıların uygulamalarda bulunan açıklıkları istismar ederek başka sitelerde betik çalıştırmasına Cross Site Scripting (siteler ötesi betik çalıştırma) denmiştir.

- **Yetkilendirme**

Kullanıcı adı ve parola doğrulaması sağlanan kullanıcıların sisteme, programa veya ağa hangi yetkilerle erişim hakkına sahip olduklarını belirten sistemdir. Sisteme kayıtlı olan kullanıcılar gruplanarak, bu gruplara çeşitli yetkiler verilir. Kullanıcı içerisinde bulunduğu grubun bütün yetkilerine sahiptir. Eğer bir kullanıcı birden fazla gruba üye ise bu gruplara verilen yetkilerin hepsine sahiptir. Güvenliğin tam olarak sağlanabilmesi için kullanıcılara gerekenden fazla yetki verilmemelidir.

- **Yığın Görevleri(Batch Jobs)**

Bilgisayarda, el ile(manuel) müdahale olmadan otomatik olarak çalışan işlemlerdir.

- **Yük Dengeleyici(Load Balancer)**

Belli bir sunucu grubuna gelen toplam bağlantı isteklerinin önceden belirlenen yöntemler ışığında bu sunuculara ileterek istekleri daha rahat karşılayacak ve gerektiğinde belleğe alabilecek sistemlerdir.