29/07/2021

OWASP Mobil TOP 10

Hazırlayan

Ümmü Derya Çelik

OWASP Nedir?

OWASP(Open Web Application Security Project) açılımı açık web uygulaması güvenlik projesi anlamına gelen, alanında özgürce edinilebilen makaleler, metodolojiler, belgeler, araçlar ve teknolojiler üreten, kar amacı gütmeyen online bir topluluktur.

OWASP Mobil Güvenlik Projesi

OWASP, en iyi web uygulamaları içindeki güvenlik açıklarına ilişkin öngörüler yayınlamasıyla tanınırken, topluluğun ayrıca mobil uygulama endüstrisine odaklanan mobil güvenlik projesidir.

Mobil Güvenlik Projesi, geliştiricilere ve güvenlik ekiplerine güvenli mobil uygulamalar oluşturmak ve sürdürmek için ihtiyaç duydukları kaynakları vermeyi amaçlayan merkezi bir kaynaktır. Projenin hedefi, mobil güvenlik risklerini sınıflandırmak ve bunların etkisini veya istismar olasılığını azaltmak için gelişimsel kontroller sağlamaktır.

Yine bu hedefler ışığında oluşturulmuş OWASP Mobil Top 10, mobil güvenlik projesini inceleyelim.

OWASP Mobil Top 10

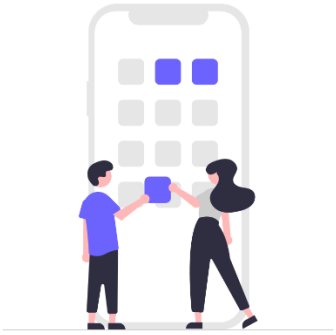
OWASP Top 10, geliştiriciler ve web uygulaması güvenliği için standart bir farkındalık belgesidir. Web uygulamalarına yönelik en kritik güvenlik riskleri hakkında geniş bir fikir birliğini temsil eder. Bu belge, dünyanın her yerinden güvenlik uzmanlarından oluşan bir ekip tarafından düzenli olarak güncellenmektedir.

OWASP üyeleri, bu belgenin şirketler tarafından benimsenmesi gerektiğini böylece yazılım geliştirme kültürünü daha güvenli kod üreten bir kültüre dönüştürmekte etkili bir adım olacağını belirtmektedirler.

En Önemli 10 Mobil Risk - 2016 Listesi

1) Hatalı Platform Kullanımı (Improper Platform Usage)

Bu kategori, bir platform özelliğinin kötüye kullanımını veya platform güvenlik kontrollerinin kullanılmamasını kapsar.



Android Manifest dosyasında yani uygulamanın erişim haklarının tanımlandığı dosyada hatalı tanımlamalar bulunan uygulamaların kullanılması, TouchID'nin kötüye kullanımı gibi eylemleri içerir. Saldırgan bu güvenlik açığını kullanarak siteler arası komut dosyası çalıştırma yani XSS saldırıları gerçekleştirebilir.

Mobil uygulamanın bu riski yaşamasının birkaç sebebi şunlardır;

a. Yayınlanmış Yönergelerin İhlali: tüm platformlarda güvenlik için geliştirme yönergeleri vardır. Bir uygulama, üretici tarafından önerilen en iyi uygulamalarla çelişirse bu riske maruz kalacaktır. Örneğin, iOS Anahtar Zincirinin nasıl kullanılacağına veya Android'de dışa aktarılan hizmetlerin nasıl güvence altına alınacağına ilişkin yönergeler vardır. Bu yönergelere uymayan uygulamalar bu riski yaşayacaktır.
b. Sözleşmenin veya genel uygulamanın ihlali
c. Kasıtsız Kötüye Kullanım: Bazı uygulamalar doğru olanı yapmayı amaçlar ancak kodlamada yapılmış bir hata buna neden olabilir. Bu, bir API çağrısında yanlış bayrağı ayarlamak gibi basit bir hata olabilir.

Bu açığı önlemek için mobil uygulamanın sunucu tarafında güvenli kodlama ve yapılandırma uygulamaları kullanılmalıdır.

2) Güvensiz Veri Depolama (Insecure Data Storage)

Yazılımcının, istemci tarafında önemli verileri tutmasıyla oluşan açıktır. Timeout süresi olmayan bir cookie istemci tarafına yazılırsa bir saldırgan buna erişip faydalanabilir.

Cihaz üzerinde tutulan log dosyaları, xml dosyaları, ikili veri depoları, SQL veri tabanı verileri istemcide erişilebilir olursa güvensiz veri kategorisine girer.

Bu açığı önlemek için, uygulamanın işlediği verilerin nasıl ele alındığına dikkat etmek gerekir. Özellikle dikkat edilmesi gereken veriler;

- URL istek ve yanıtlarını önbelleğe alma
- Kullanıcının klavye verilerini önbelleğe alma
- Kopyala/yapıştır verilerini önbelleğe alma
- Log kayıtları
- Üçüncü taraflara gönderilen analiz verileri

3) Güvensiz Haberleşme (Insecure Communication)

Bir mobil uygulama tasarlarken, veriler istemci ve sunucu arasında iletilir. Bu esnada veri, mobil cihazın operatör ağını ve interneti geçmelidir. Saldırgan, internet ağı boyunca ilerleyen verileri ele geçirmek için güvenlik açıklarından faydalanabilir.

Saldıran, yerel ağınızı paylaşma(güvenliği ihlal edilmiş veya izlenen WiFi), yönlendiriciler veya ağ cihazlarını kullanma, mobil cihazınıza yüklü kötü amaçlı yazılım sayesinde verilerinizi ele geçirebilir. Ortadaki adam saldırıları bunlara örnektir.

Mobil uygulamalar genellikle ağ trafiğini korumaz. Kimlik doğrulama sırasında SSL/TLS kullanabilirler ancak başka yerde kullanamazlar. Bu tutarsızlık, verilerin ve oturum kimliklerinin müdahaleye maruz kalması riskine yol açar. Temel kusurları tespit etmek için telefonun ağ trafiği gözlemlenebilir.

Ayrıca https yerine http kullanmak ya da https'nin eski sürümlerinin kullanılması da iletişimi güvensiz hale getirecektir.

Bu risk, bir mobil cihazın kullanabileceği tüm iletişim teknolojilerini içerir: TCP/IP, WiFi, Bluetooth/Bluetooth-LE, NFC, ses, kızılötesi, GSM, 3G, SMS gibi.

Bu riskin oluşması için belirleyici özellik, iki cihazın varlığı ve aralarında veri akışı olmasıdır.

Bu açığı önlemek için şunlar uygulanabilir;

- Mobil uygulamanın hassas bilgileri, oturum belirteçlerini veya diğer hassas verileri bir arka uç API'sine veya web hizmetine iletmek için kullanacağı aktarım kanallarına SSL/TLS uygulanmalıdır.

- Kullanıcının oturum kimliğini açığa çıkarabileceğinden, karışık SSL oturumlarından kaçılmalıdır.

- Uygun anahtar uzunluklarına sahip güçlü, endüstri standardı şifre paketleri kullanılmalıdır.

- Sertifika Yetkilisi tarafından imzalanmış güvenilir sertifikalar kullanılmalıdır.

- Kendinden imzalı sertifikalara izin verilmemelidir.

- Hassas veriler alternatif kanallar üzerinden gönderilmemelidir. SMS, MMS gibi.

- SSL kanalına verilmeden önce tüm hassas verilere ayrı bir şifreleme katmanı uygulanabilir böylece SSL'deki olağan güvenlik açıklarına karşı şifreli veriler ikincil korumaya sahip olacaktır.

Pentester'lar tarafından en sık karşılaşılan iletişim güvenliği senaryoları sertifika denetimi eksikliği, zayıf Handshake işlemi, gizlilik bilgisi sızıntısı şeklindedir.

4) Güvensiz Kimlik Doğrulama (Insecure Authentication)

Uygulamada, erişim belirteci sağlamadan arka uç API hizmet isteği anonim olarak çalıştırılabiliyorsa yani yapılandırma doğru olmadığı için aktivite baypaslanarak müdahale edinilebiliyorsa, doğrudan web servis isteklerini Burpsuit ile ya da bunun gibi ağ dinleme aracı ile dinleyerek API hizmet isteği ele geçirilebiliyorsa güvensiz kimlik doğrulama kategorisine girmektedir.

Güvensiz 2FA(two factor authentication) yani iki faktörlü kimlik doğrulama da kullanıcı adı ve parola girildiğinde cihaza SMS ile OTP yani tek kullanımlık şifre gelir. OTP normal şartlarda kullanıcıdan başka kimse tarafından erişilemez olmalıdır fakat aynı kullanıcı adı ve şifreyi giren saldırgan http response' unda açık şekilde SMS içeriğini görebilir.

Yine OTP baypaslarında en çok kullanılan açık, yanlış veya doğru otp girildiğinde true ya da false değer döner. Saldırgan bir değer girer false değeri döndüğünde burpsuit gibi araçlarla mobil cihaza true değer olan OTP gönderilmeden isteği kesip müdahale ederek false dönen değeri true' ya çevirip kendi girdi değerin mobil cihaza gönderilmesini sağlayarak yine başarılı şekilde giriş yapmış olur. En sık görülen kimlik doğrulama açıklarındandır.

Bu açığı önlemek için;

- Bir web uygulamasını mobil eşdeğerine taşınıyorsa, mobil uygulamaların kimlik doğrulama gereksinimleri, web uygulaması bileşenininkiyle eşleşmelidir.

- Tüm kimlik doğrulama isteklerinin sunucu tarafında gerçekleştirildiğinden emin olunmalıdır. Başarılı kimlik doğrulamanın ardından, uygulama verileri mobil cihaza yüklenecektir. Bu, uygulama verilerinin yalnızca başarılı kimlik doğrulamasından sonra kullanılabilir olmasını sağlayacaktır.

- Verilerin istemci tarafında depolanması gerekiyorsa, verilerin, kullanıcının oturum açma kimlik bilgilerinden türetilen bir şifreleme anahtarı kullanılarak şifrelenmesi gerekir.

- Mobil uygulamalarda uygulanan kalıcı kimlik doğrulama (Beni Hatırla) işlevi, hiçbir zaman bir kullanıcının parolasını cihazda saklamamalıdır.

- Bir kullanıcının kimliğini doğrulamak için yanıltıcı değerler kullanılmamalıdır. Buna cihaz tanımlayıcıları veya coğrafi konum dahildir.

- Mümkünse kullanıcıların kimlik doğrulama parolaları için dört haneli PIN numaraları kullanmasına izin verilmemelidir.

- Geliştiriciler, tüm istemci tarafı yetkilendirme ve kimlik doğrulama kontrollerinin kötü niyetli kullanıcılar tarafından atlanabileceğini varsaymalıdır. Yetkilendirme ve kimlik doğrulama kontrolleri, mümkün olduğunda sunucu tarafında yeniden uygulanmalıdır.

5) Yetersiz Kriptografi (Insufficient Cryptography)

Uygun olmayan şekilde şifrelenmiş veriler bu kategoriye dahil olabilecek açıklara sebep olmaktadır. En yaygın görülen hatalar, yanlış şifreleme, zayıf şifreler, Encryption ve encoding kavramlarının karıştırılmasıyla oluşmaktadır.

Encryption ile Encoding kavramlarını yazılımcılar karıştırdığı takdirde uygulamada büyük bir açık oluşur. Encryption kavramında bir ya da iki key yani anahtar yardımıyla şifreleme yapılır. Encrypt edilmiş yani şifrelenmiş bir veri anahtar olmadan açılamaz ve bu şekilde veri güvenliğini sağlar. Fakat yazılımcı şifreleme yaptığını düşünürken evrensel standartlara sahip, çözülmesi çok kolay olan encoding kullanarak şifrelerse saldırgan decoding ederek çok kolay bir şekilde verilere erişebilir.

Her zaman güvenlik topluluğu tarafından güçlü olarak kabul edilen modern algoritmalar kullanılmalıdır ve mümkün olduğunda mobil platformda son teknoloji şifreleme API' lerinden yararlanılmalıdır.

RC2, MD4, MD5, SHA1 gibi algoritma ve protokoller önemli açıkları olduğu veya modern güvenlik gereksinimlerini karşılamadığı için kullanılması önerilmemektedir.

6) Güvensiz Yetkilendirme (Insecure Authorization)

Önemli bir zafiyet olan IDOR zafiyeti de bu kategoride yer alır. IDOR güvensiz nesnelere yönelim olarak Türkçeye çevrilebilir. Bu zafiyetin işleyişi şu şekildedir;

Mobil cihazdan bir web sitesi ziyaret edildiğinde içeriğinde bulunan uygulamalara nesneler üzerinden erişim sağlanır. Bu nesneler; veri tabanı, dosyaları ve dizinlere erişim gibi önemli durumları da tanımlamakta kullanılmaktadır.

Saldırganlar bir başka kullanıcının sahip olduğu nesne değerlerini taklit veya manipüle edebilmektedir. Böylelikle hedeflediği kişinin uygulama üzerindeki kimlik bilgilerini elde etmiş olurlar. Saldırganlar tarafından ele geçirilen bir hesap üzerindeki tüm bilgiler kontrol edilebilmektedir.

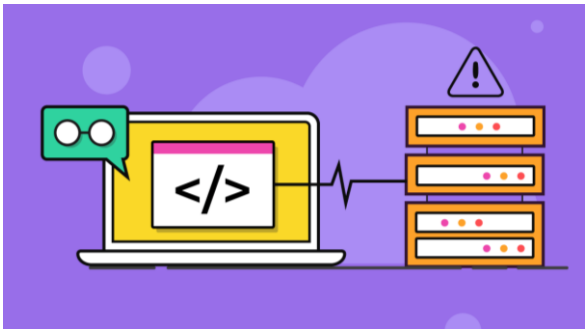
Ayrıca IDOR zafiyeti kimlik doğrulama kontrollerinin atlatılmasını mümkün kılacağından dolayı, saldırganların bu durumu kötüye kullanarak daha yetkili bir hesaba erişim sağlamaları da mümkün olacaktır.

Yetkili hesaba erişim sağlayan saldırgan, sistemde tespit edilen diğer zafiyetleri de kullanarak zincirleme zafiyet istismarları gerçekleştirebilir. Bu yöntem sızma testi çalışmaları esnasında siber güvenlik uzmanları tarafından yetki yükseltme (privilege escalation) aşaması dahilinde de kullanılmaktadır.

Bu riski önlemek için arka uç sistemlerinde bulunan bilgileri kullanarak kimliği doğrulanmış kullanıcının rolleri ve izinleri doğrulanmalıdır. Mobil cihazın kendisinden gelen herhangi bir role veya izin bilgisine güvenmekten kaçınılmalıdır.

7) İstemci Kod Kalite Sorunları (Client Code Quality)

Sunucu tarafında değil istemci tarafında yapılan kodlama hatalarından dolayı oluşan zafiyetler bu kategoriye dahil olmaktadır.



İstemci tarafı kodlarında üçüncü parti kütüphanelerinin kullanılması, uygulamada Buffer overflow yani ara bellek taşması görülmesi, Format string yani kullanıcının belirtilenden fazla veya farklı stringler girmesine izin verilmesi bu zafiyetlere örnektir.

Tehdit Aracıları, mobil kod içinde yapılan yöntem çağrılarına güvenilmeyen girdileri iletebilen varlıkları içerir. Bu tür sorunlar, kendi başlarına güvenlik sorunları olmak zorunda değildir, ancak güvenlik açıklarına yol açar.

Örneğin, Safari'nin eski sürümlerinde arabellek taşmaları (düşük kod kalitesi güvenlik açığı), yüksek riskli Jailbreak saldırılarına yol açtı.

Düşük kod kalitesi sorunlarından genellikle kötü amaçlı yazılım veya kimlik avı dolandırıcılığı yoluyla yararlanılır.

Bu zafiyeti önlemek için;

- Kuruluşta herkesin üzerinde anlaştığı tutarlı kodlama kalıpları kullanılmalıdır.

- Okunması kolay ve iyi belgelenmiş kod yazılmasına dikkat edilmelidir.

- Buffer kullanırken, gelen arabellek verilerinin uzunluklarının hedef arabellek uzunluğunu aşmayacağı doğrulanmalıdır.

- Otomasyon aracılığıyla, üçüncü taraf statik analiz araçlarını kullanarak arabellek taşmalarını ve bellek sızıntılarını belirlenmelidir.

- Arabellek taşmalarını ve bellek sızıntılarını çözmeye öncelik verilmelidir.

8) Kod Kurcalama (Code Tampering)

Tipik olarak, bir saldırgan, üçüncü taraf uygulama mağazalarında barındırılan uygulamaların kötü niyetli biçimleri aracılığıyla kod değişikliğinden yararlanır.

Uygulama mobil cihaza yüklendikten sonra kod ve veri kaynakları da beraberinde gelir. Saldırgan apk dosyalarını ".smali" dosyalarına dönüştürüp koda zararlı kodlar, fonksiyonlar veya izinler ekleyebilir. Uygulama çalışma anında Frida gibi canlı analiz yapan araçlarla araya girip API çağrılarına müdahale edip yine çeşitli izinler ve yetkiler verebilir. Böylece saldırgan bu açıklardan faydalanarak kişisel ya da parasal kazanç için yazılımın kullanım amacını yıkıp kötü amaçlı kullanabilir.

Teknik olarak, tüm mobil kodlar, kod kurcalamaya karşı savunmasızdır. Mobil kod, kodu üreten kuruluşun kontrolünde olmayan bir ortamda çalışır. Aynı zamanda, bu kodun çalıştığı ortamı değiştirmenin birçok farklı yolu vardır. Bu değişiklikler, bir rakibin kodu kurcalamasına ve istediği zaman değiştirmesine izin verir.

Bu riski önleyebilmek için mobil uygulama, derleme zamanında bütünlüğü hakkında bildiklerinden kodun eklendiğini veya değiştirildiğini çalışma

zamanında algılayabilecek, çalışma zamanında bir kod bütünlüğü ihlaline uygun şekilde tepki verebilecek şekilde yazılmalıdır.

9) Tersine Mühendislik (Reverse Engineering)

Temel mantık, saldırgan hedeflenen uygulamayı bir uygulama mağazasından indirir ve bir dizi farklı araç kullanarak kendi yerel ortamında analiz eder. Saldırganın amacı ise uygulamanın kaynak kodu, kütüphaneleri, algoritması ve diğer kaynakları tespit ederek doğabilecek açıklar, parolalar, anahtarlar gibi özel bilgileri yararına kullanmaktır.

Saldırganın uygulamayı inceleyebilmek için kaynak kodlara ihtiyacı olacaktır. Kaynak koda erişmek için uygulama dosyasını yani APK'yi decompile eder yani kodu derleyerek kaynak koda dönüştürür. Bu durumda saldırgan kaynak kodu görebilmektedir fakat decompile edilmiş APK dosyasının üzerinde herhangi bir değişiklik, fonksiyon ekleme, çıkarma gibi işlemler yapılamaz.

Buna rağmen yine de apk kaynak kodlarında değişiklik yapıp kodu çalıştırmanın bir yolu mevcut. O da apk dex dosyalarını yani decompile edilmiş dosyaları ".smali" uzantılı dosyalara dönüştürmektir. Bu işlem Apktool gibi birçok araçtan yapılabilir.

Smali dosyaları decompile edilmiş kodlardan daha karmaşık gözükür. Bunun nedeni bu kategoriye de dahil olan olası saldırıları önlemek için kod karmaşıklaştırma yöntemi kullanılmasıdır.

Kod karmaşıklaştırma işlemi, uygulama piyasaya sürülmeden önce kaynak kodunda fonksiyonların ve sınıfların adlarının anlamsız şekilde değiştirilerek saldırgan ve ya herhangi bir kişi tarafından kolaylıkla anlamasını önleme işlemidir. Fakat yine kod karmaşıklaştırma işlemini geri döndürmenin de yolu vardır. Simplify gibi araçlar karmaşıklaştırılmış kaynak kodu aslına dönüştürülebilir.

Bu riski azaltabilmek için kod karmaşıklaştırma işlemi yapan Obfuscator aracı şu özelliklere sahip olmalıdır;

- Hangi yöntemlerin/kod bölümlerinin karmaşıklaştırılacağı belirlenmelidir.

- Karmaşıklaştırma işlemi derecesi uygulamanın performansını etkilemeyecek şekilde olmalıdır.

- Simplify, IDA Pro, Hopper gibi araçlardan gelen saldırılara dayanıklı olmalıdır.

- Metotların yanı sıra dize tablolarını gizlenmelidir.

10) Gereksiz İşlevsellik (Extraneous Functionality)

Genellikle geliştiricilerin geliştirme sırasında unuttuğu fonksiyonları içermektedir. Uygulama yayınlanmadan önce test aşamasında 2FA girmemek için disable edilmiş yani devre dışı bırakılmış bir özellik unutulup uygulama yayınlandığında kimlik doğrulamayı atlama gibi güvenlik açıkları oluşturur.

Tipik olarak, bir saldırgan, arka uç sistemlerdeki açığı keşfetmek için bir mobil uygulama içindeki gereksiz işlevleri anlamaya çalışır. Saldırgan, son kullanıcıya herhangi bir müdahale de bulunmadan, doğrudan kendi sistemlerinden bulduğu gereksiz işlevsellerden yararlanır.

Herhangi bir mobil uygulamanın, arayüz aracılığıyla doğrudan kullanıcıya açıklanmayan gereksiz işlevsellik içerme olasılığı yüksektir. Bu ek kodun çoğu, doğası gereği zararsızdır ve saldırgana arka uç yetenekleri hakkında herhangi bir ek bilgi vermez. Ancak, bazı gereksiz işlevler bir saldırgan için çok yararlı olabilir. Arka uç testi, demo, hazırlama veya UAT ortamlarıyla ilgili bilgileri ortaya çıkaran işlevsellik, bir üretim yapısına dahil edilmemelidir.

Bu güvenlik açığını önlemenin en iyi yolu kod hakkında en bilgili konu uzmanlarını kullanarak manuel güvenli kod incelemesi yapmaktır.

Alınabilecek önlemler ise şu şekildedir;

- Gizli anahtarları keşfetmek için uygulamanın yapılandırma ayarları incelenmelidir.

- Tüm test kodunun uygulamanın son üretim derlemesine dahil edilmediğini doğrulanmalıdır. Test etmek için eklenmiş fonksiyonlar gibi kodlar.

- API uç noktalarının iyi belgelenip belgelenmediği incelenmelidir.

- Log kayıtlarına arka uç hakkında aşırı açıklayıcı hiçbir şey yazılmadığından emin olmak için tüm kayıtlar incelenmelidir.