Uygulama Güvenliği Doğrulama Standardı

Versiyon 5.0.0

Mayıs 2025

Contents

Bilgilendirme

Standart Hakkında

Uygulama Güvenliği Doğrulama Standardı, mimarlar, geliştiriciler, test uzmanları, güvenlik profesyonelleri, araç satıcıları ve tüketiciler tarafından güvenli uygulamaları tanımlamak, inşa etmek, test etmek ve doğrulamak için kullanılabilecek uygulama güvenliği gereksinimleri veya testlerinin bir listesidir.

Telif Hakkı ve Lisans

Versiyon 5.0.0, Mayıs 2025

Figure 1: license

Proje Liderleri

Elar Lang	Josh C Grossman
Jim Manico	Daniel Cuthbert

Çalışma Grubu

Tobias Ahnoff	Ralph Andalis	Ryan Armstrong	Gabriel Corona
Meghan Jacquot	Shanni Prutchi	Iman Sharafaldin	Eden Yardeni

Diğer Önemli Katkıda Bulunanlar

Sjoerd Langkemper	Isaac Lewis

Mark Carney	Sandro Gauci

Diğer Katkıda Bulunanlar ve Değerlendirenler

Diğer katkıda bulunanların bir listesini Ek E'ye ekledik.

Eğer 5.x katkıda bulunanlar listesinde eksik bir kişi varsa, gelecekteki 5.x güncellemelerinde eklenmesi için lütfen GitHub'da bir olay açın.

Uygulama Güvenliği Doğrulama Standardı, 2008'de ASVS 1.0'dan 2019'da ASVS 4.0'a kadar katkıda bulunanların emeğiyle inşa edilmiştir. ASVS'nin bugünkü yapısının ve doğrulama maddelerinin çoğu, başlangıçta Andrew van der Stock, Mike Boberski, Jeff Williams ve/veya Dave Wichers tarafından yazılmış olsa da, katkıda bulunan çok daha fazla kişi bulunmaktadır. Daha önce katkıda bulunan herkese teşekkür ederiz. Önceki sürümlere katkıda bulunanların tam listesi için lütfen önceki ilgili sürümleri inceleyin.

Önsöz

Uygulama Güvenliği Doğrulama Standardı (ASVS) 5.0 sürümüne hoş geldiniz.

Giriş

İlk olarak 2008 yılında küresel bir topluluk iş birliğiyle başlatılan ASVS, modern web uygulamaları ve servislerinin tasarımı, geliştirilmesi ve test edilmesi için kapsamlı bir güvenlik gereksinimleri setini tanımlar.

2019'da ASVS 4.0 sürümü ve 2021'deki küçük güncellemesi (v4.0.3) sonrası, 5.0 sürümü yazılım güvenliğindeki en son gelişmeleri yansıtacak şekilde modernleştirilmiş önemli bir dönüm noktasıdır.

ASVS 5.0, proje liderleri, çalışma grubu üyeleri ve bunlardan daha geniş OWASP topluluğunun bu önemli standardı güncellemek ve geliştirmek amacıyla verdikleri yoğun katkıların sonucudur.

5.0 Sürümünün Arkasındaki İlkeler

Bu büyük revizyon, birkaç temel ilke göz önünde bulundurularak geliştirilmiştir:

- Daha Net Kapsam ve Odak: Standardın bu sürümü, adındaki temel dayanaklar "Uygulama, Güvenlik, Doğrulama ve Standart"ile daha doğrudan uyumlu olacak şekilde tasarlanmıştır. Gereksinimler, belirli teknik uygulamaları zorunlu kılmak yerine, güvenlik açıklarının önlenmesine vurgu yapacak şekilde yeniden yazılmıştır. Gereksinim metinlerinin kendi kendilerini açıklaması ve neden var olduklarını açıklaması amaçlanmıştır.

- Belgelenmiş Güvenlik Kararları Desteği: ASVS 5.0, temel güvenlik kararlarının belgelenmesine dair gereksinimler getirmektedir. Bu sayede izlenebilirlik artar ve bağlama duyarlı uygulamalar desteklenerek, organizasyonların güvenlik duruşlarını özel ihtiyaç ve risklerine göre uyarlamalarına olanak sağlar.
- Güncellenmiş Seviyeler: ASVS üç katmanlı modelini korurken, benimsenmelerinin kolaylaştırması için seviye tanımları geliştirilmiştir. Seviye 1, ASVS'yi benimsemenin ilk adımı olarak tasarlanmıştır ve ilk savunma katmanını sağlamaktadır. Seviye 2, standart güvenlik uygulamalarının kapsamlı bir görünümünü sunar. Seviye 3 ise gelişmiş ve yüksek seviye güvence gereksinimlerini kapsar.
- Yeniden Yapılandırılmış ve Genişletilmiş İçerik: ASVS 5.0, 17 bölümde yaklaşık 350 gereksinimi içermektedir. Bölümler, açıklık ve kullanılabilirliği artırmak için yeniden düzenlenmiştir. Geçişi kolaylaştırmak amacıyla v4.0 ile v5.0 arasında çift yönlü bir eşleştirme sağlanmıştır.

ASVS'nin Geleceği

Bir uygulamanın güvenliğini sağlamak hiçbir zaman tam anlamıyla bitmeyeceği gibi, ASVS de bitmeyecektir. 5.0 sürümü büyük bir adım olsa da, geliştirme süreci devam etmektedir. Bu sürüm, topluluğun biriken geliştirmelerden ve eklemelerden faydalanmasını sağlamakla kalmaz, aynı zamanda gelecekteki iyileştirmeler için zemin hazırlar. Bunlar arasında topluluk tarafından oluşturulan uygulama ve doğrulama rehberlerinin temel gereksinim seti üzerine inşa edilmesi de olabilir.

ASVS 5.0, güvenli yazılım geliştirme için güvenilir bir temel oluşturacak şekilde tasarlanmıştır. Topluluk, uygulama güvenliğinin durumunu kolektif olarak ilerletmek için bu standardı benimsemeye, katkıda bulunmaya ve üzerine inşa etmeye davet edilmektedir.

ASVS Nedir?

Uygulama Güvenliği Doğrulama Standardı (ASVS), web uygulamaları ve hizmetleri için güvenlik gereksinimlerini tanımlar ve güvenli uygulamalar tasarlamayı, geliştirmeyi ve sürdürmeyi veya bunların güvenliğini değerlendirmeyi amaçlayan herkes için değerli bir kaynaktır.

Bu bölümde ASVS'nin kapsamı, önceliğe dayalı seviyelerinin yapısı ve standart için birincil kullanım durumları da dahil olmak üzere ASVS'yi kullanmanın temel yönleri özetlenmektedir.

ASVS'nin Kapsamı

ASVS'nin kapsamı adıyla tanımlanmaktadır: Uygulama, Güvenlik, Doğrulama ve Standart. Ulaşılması gereken güvenlik ilkelerini tanımlamak amacıyla hangi gereksinimlerin dahil edileceğini veya hariç tutulacağını belirler. Kapsam, uygulama gereksinimleri için temel teşkil eden dokümantasyon gereksinimlerini de dikkate alır.

Saldırganlar için kapsam diye bir şey yoktur. Bu nedenle ASVS gereksinimleri, CI/CD süreçleri, hosting ve operasyonel faaliyetler de dahil olmak üzere uygulama yaşam döngüsünün diğer yönlerine ilişkin yönlendirmelerle birlikte değerlendirilmelidir.

Uygulama

ASVS bir "uygulamayı", güvenlik kontrollerinin entegre edilmesi gereken ve geliştirilmekte olan yazılım ürünü olarak tanımlar. ASVS, geliştirme yaşam döngüsü faaliyetlerini belirlemez veya uygulamanın bir CI/CD boru hattı aracılığıyla nasıl oluşturulması gerektiğini dikte etmez. Bunun yerine, ürünün kendi içinde elde edilmesi gereken güvenlik sonuçlarını belirtir.

Web Uygulaması Güvenlik Duvarları (WAF'lar), yük dengeleyiciler veya proxy'ler gibi HTTP trafiğini sunan, değiştiren veya doğrulayan bileşenler, bazı güvenlik kontrolleri doğrudan bunlara bağlı olduğundan veya bunlar aracılığıyla uygulanabildiğinden, bu özel amaçlar için uygulamanın bir parçası olarak kabul edilebilir. Bu bileşenler, önbelleğe alınan yanıtlar, hız sınırlaması veya gelen ve giden bağlantıların kaynak ve hedefe göre kısıtlanmasıyla ilgili gereksinimler için dikkate alınmalıdır.

Tersine, ASVS genellikle uygulamayla doğrudan ilgili olmayan veya yapılandırmanın uygulamanın sorumluluğu dışında olduğu gereksinimleri hariç tutar. Örneğin, DNS sorunları genellikle ayrı bir ekip veya işlev tarafından yönetilir.

Benzer şekilde, uygulama girdiyi nasıl tükettiğinden ve çıktıyı nasıl ürettiğinden sorumlu olsa da, harici bir süreç uygulama veya verileriyle etkileşime giriyorsa, ASVS için kapsam dışı kabul edilir. Örneğin, uygulamanın veya verilerinin yedeklenmesi genellikle harici bir sürecin sorumluluğundadır ve uygulama veya geliştiricileri tarafından kontrol edilmez.

Güvenlik

Her gerekliliğin güvenlik üzerinde kanıtlanabilir bir etkisi olmalıdır. Bir gerekliliğin olmaması daha az güvenli bir uygulama ile sonuçlanmalı ve gerekliliğin uygulanması bir güvenlik riskinin olasılığını veya etkisini azaltmalıdır.

İşlevsel yönler, kod stili veya politika gereksinimleri gibi diğer tüm hususlar kapsam dışıdır.

Doğrulama

Gereksinim doğrulanabilir olmalı ve doğrulama "başarısız"veya "başarılı"kararıyla sonuçlanmalıdır.

Standart

ASVS, standarda uymak için uygulanması gereken bir güvenlik gereksinimleri koleksiyonu olarak tasarlanmıştır. Bu, gereksinimlerin bunu başarmak için güvenlik hedefini tanımlamakla sınırlı

olduğu anlamına gelir. Diğer ilgili bilgiler ASVS'nin üzerine inşa edilebilir veya eşleştirmeler yoluyla bağlanabilir.

Özellikle, OWASP'ın birçok projesi vardır ve ASVS kasıtlı olarak diğer projelerdeki içerikle örtüşmekten kaçınır. Örneğin, geliştiricilerin "belirli bir gereksinimi kendi teknolojimde veya ortamımda nasıl uygulayabilirim"şeklinde bir sorusu olabilir ve bu, Cheat Sheet Series projesi kapsamında ele alınmalıdır. Doğrulayıcıların "bu gereksinimi bu ortamda nasıl test edebilirim"şeklinde bir sorusu olabilir ve bu Web Güvenliği Test Kılavuzu projesi kapsamında ele alınmalıdır.

ASVS sadece güvenlik uzmanlarının kullanımına yönelik olmamakla birlikte, okuyucunun içeriği anlamak için teknik bilgiye veya belirli kavramları araştırma becerisine sahip olmasını beklemektedir.

Gereksinim

Gereksinim kelimesi ASVS'de özel olarak kullanılır çünkü bu kelimenin karşılanması için nelerin başarılması gerektiğini tanımlar. ASVS sadece gereksinimleri (must) içerir ve ana koşul olarak tavsiyeleri (should) içermez.

Başka bir deyişle, ister bir sorunu çözmek için birçok olası seçenekten biri isterse de kod stili hususları olsun, öneriler bir gereksinim olma tanımını karşılamaz.

ASVS gereksinimleri, uygulamaya veya teknolojiye özgü olmadan belirli güvenlik ilkelerini ele almayı ve aynı zamanda neden var olduklarına dair açıklayıcı olmayı amaçlamaktadır. Bu aynı zamanda gereksinimlerin belirli bir doğrulama yöntemi veya uygulaması etrafında oluşturulmadığı anlamına gelir.

Belgelenmiş güvenlik kararları

Yazılım güvenliğinde, güvenlik tasarımını ve kullanılacak mekanizmaları erkenden planlamak, bitmiş üründe veya özellikte daha tutarlı ve güvenilir bir uygulama oluşturacaktır.

Ayrıca, belirli gereksinimler için uygulama, karmaşık ve bir uygulamanın ihtiyaçlarına çok spesifik olacaktır. Yaygın örnekler arasında izinler, girdi doğrulama ve farklı hassas veri seviyeleri etrafında koruyucu kontroller yer alır.

Bunu hesaba katmak için "tüm veriler şifrelenmelidir"gibi kapsamlı ifadeler veya bir gereksinimde olası her kullanım durumunu kapsamaya çalışmak yerine, uygulama geliştiricisinin bu tür kontrollere yaklaşımının ve yapılandırmasının belgelenmesini zorunlu kılan dokümantasyon gereksinimleri dahil edilmiştir. Bu daha sonra uygunluk açısından gözden geçirilebilir ve ardından gerçek uygulama, uygulamanın beklentilere uyup uymadığını değerlendirmek için belgelerle karşılaştırılabilir.

Bu gereksinimler, uygulamayı geliştiren kuruluşun belirli güvenlik gereksinimlerinin nasıl uygulanacağına ilişkin aldığı kararları belgelemeyi amaçlamaktadır.

Dokümantasyon gereksinimleri her zaman bir bölümün ilk kısmında yer alır (her bölümde olmasa da) ve her zaman dokümante edilen kararların gerçekten uygulamaya konulması gereken ilgili bir uygulama gerekliliğine sahiptir. Burada önemli olan nokta, dokümantasyonun yerinde olduğunu ve fiili uygulamanın iki ayrı faaliyet olduğunu doğrulamaktır.

Bu gereksinimleri dahil etmenin iki temel nedeni vardır. İlk neden, güvenlik gereksiniminin genellikle kuralların uygulanmasını içermesidir. Örneğin hangi tür dosya türlerinin yüklenmesine izin verilir, hangi iş kontrolleri uygulanmalıdır, belirli bir alan için izin verilen karakterler nelerdir gibi uygulanma biçimleri. Bu kurallar her uygulama için farklı olacaktır ve bu nedenle ASVS, bunların ne olması gerektiğini kuralcı bir şekilde tanımlayamaz ve bu durumda bir cheat sheet veya daha ayrıntılı bir yanıt da yardımcı olmaz. Benzer şekilde, bu kararlar belgelenmeden, bu kararları uygulayan gereksinimlerin doğrulanması mümkün olmayacaktır.

İkinci neden ise, belirli gereksinimler için belirli güvenlik sorunlarının nasıl ele alınacağı konusunda esnekliğe sahip bir uygulama geliştirme sağlamanın önemli olmasıdır. Örneğin, önceki ASVS sürümlerinde oturum zaman aşımı kuralları çok kuralcıydı. Pratik olarak, birçok uygulama, özellikle tüketicilere yönelik olanlar, çok daha esnek kurallara sahiptir ve bunun yerine diğer risk azaltma kontrollerini uygulamayı tercih eder. Bu nedenle, belgeleme gereksinimleri bu konuda açıkça esneklik sağlar.

Elbette ki, bu kararların bireysel geliştiriciler tarafından alınması ve belgelenmesi beklenmemektedir. Aksine, bu kararlar organizasyonun bir bütün olarak alınacak ve geliştiricilere iletilecek, geliştiriciler de bu kararları uygulamaya özen gösterecektir.

Geliştiricilere yeni özellikler ve işlevler için spesifikasyonlar ve tasarımlar sağlamak, yazılım geliştirmenin standart bir parçasıdır. Benzer şekilde, geliştiricilerin her seferinde kendi kararlarını vermek yerine ortak bileşenleri ve kullanıcı arayüzü mekanizmalarını kullanmaları beklenir. Bu nedenle, bunu güvenliğe genişletmek şaşırtıcı veya tartışmalı olarak görülmemelidir.

Bunu nasıl başaracağınız konusunda da esneklik vardır. Güvenlik kararları, geliştiricilerin başvurması beklenen yazılı bir belgede belgelenebilir. Alternatif olarak, güvenlik kararları belgelenebilir ve tüm geliştiricilerin kullanması zorunlu olan ortak bir kod kütüphanesinde uygulanabilir. Her iki durumda da istenen sonuç elde edilir.

Uygulama Güvenliği Doğrulama Seviyeleri

ASVS, seviyesi arttıkça derinliği ve karmaşıklığı artan üç güvenlik doğrulama seviyesi tanımlar. Genel amaç, kuruluşların en kritik güvenlik sorunlarını ele almak için ilk seviyeden başlaması ve ardından kuruluşun ve uygulamanın ihtiyaçlarına göre daha yüksek seviyelere geçmesidir. Seviyeler, belgede ve gereksinim metinlerinde L1, L2 ve L3 olarak gösterilebilir.

Her ASVS seviyesi, o seviyeden elde edilmesi gereken güvenlik gereksinimlerini belirtir ve kalan daha yüksek seviye gereksinimler öneri olarak sunulur.

Yinelenen gereksinimleri veya daha yüksek seviyelerde artık geçerli olmayan gereksinimleri önlemek için bazı gereksinimler belirli bir seviyeye özel uygulanır. Fakat bu gereksinimler daha yüksek

seviyeler için daha katı koşullar içerir.

Seviye değerlendirmesi

Seviyeler, güvenlik gereksinimlerinin uygulanması ve test edilmesi deneyimine dayalı olarak her bir gereksinimin öncelik bazlı değerlendirmesiyle tanımlanır. Ana odak noktası, risk azaltma ile gereksinimin uygulanması için gereken çabayı karşılaştırmaktır. Bir diğer önemli faktör ise giriş engelini düşük tutmaktır.

Risk azaltma, gereksinimin uygulama içindeki güvenlik riskinin düzeyini ne ölçüde azalttığını, klasik Gizlilik, Bütünlük ve Kullanılabilirlik etki faktörlerini dikkate alarak ve bunun birincil savunma katmanı mı yoksa derinlemesine savunma mı olduğunu değerlendirerek değerlendirir.

Kriterler ve seviyelendirme kararları etrafında yapılan titiz tartışmalar, her duruma %100 uymayabileceğini kabul etmekle birlikte, vakaların büyük çoğunluğu için geçerli olması gereken bir dağılımla sonuçlanmıştır. Bu, belirli durumlarda kuruluşların kendi özel risk değerlendirmelerine göre daha yüksek seviyedeki gereksinimleri daha erken önceliklendirmek isteyebilecekleri anlamına gelir.

Her seviyedeki gereksinim türleri aşağıdaki gibi tanımlanabilir:

Seviye 1

Bu seviye, bir uygulamayı güvenli hale getirirken dikkate alınması gereken minimum gereksinimleri içerir ve kritik bir başlangıç noktasıdır. Bu seviye, ASVS gereksinimlerinin yaklaşık %20'sini içerir. Bu seviyenin amacı, giriş engelini azaltmak için mümkün olduğunca az gereksinim olmasını sağlamaktır.

Bu gereksinimler genellikle kritik veya temel, diğer güvenlik açıklarının veya ön koşulların istismar edilmesini gerektirmeyen yaygın saldırıları önlemek için birinci katman savunma gereksinimleridir.

Birinci savunma katmanı gereksinimlerine ek olarak, bazı gereksinimlerin daha yüksek seviyelerde etkisi daha azdır, şifrelerle ilgili gereksinimler gibi. Bunlar Seviye 1 için daha önemlidir, çünkü daha yüksek seviyelerde çok faktörlü kimlik doğrulama gereksinimleri önem kazanır.

Seviye 1, belgelere veya koda iç erişimi olmayan bir dış test uzmanı tarafından mutlaka penetrasyon testi yapılabilir değildir (örneğin "kara kutu"testi), ancak gereksinimlerin sayısının az olması doğrulamayı kolaylaştırmalıdır.

Seviye 2

Çoğu uygulama bu güvenlik seviyesine ulaşmak için çaba göstermelidir. ASVS'deki gereksinimlerin yaklaşık %50'si L2 seviyesindedir, yani bir uygulamanın L2 seviyesine uymak için ASVS'deki gereksinimlerin yaklaşık %70'ini (tüm L1 ve L2 gereksinimlerini) uygulaması gerekir.

Bu gereksinimler genellikle daha az yaygın saldırılarla veya yaygın saldırılara karşı daha karmaşık korumalarla ilgilidir. Bunlar hala ilk savunma katmanı olabilir veya saldırının başarılı olması için belirli ön koşullar gerektirebilir.

Seviye 3

Bu seviye, en yüksek güvenlik seviyelerini göstermek isteyen uygulamalar için bir hedef olmalıdır. Bu seviye, uyum sağlamak için gereken gereksinimlerin son %30'unu sağlar.

Bu bölümdeki gereksinimler genellikle derinlemesine savunma mekanizmaları veya diğer yararlı ancak uygulanması zor kontrollerden oluşur.

Hangi seviyeye ulaşılmalı?

Öncelik tabanlı seviyeler, kuruluşun ve uygulamanın uygulama güvenliği olgunluğunu yansıtmayı amaçlamaktadır. ASVS, bir uygulamanın hangi seviyede olması gerektiğini kuralcı bir şekilde belirtmek yerine, kuruluşun risklerini analiz etmesi ve uygulamanın hassasiyetine ve elbette uygulamanın kullanıcılarının beklentilerine bağlı olarak hangi seviyede olması gerektiğini belirlemesi gerekir.

Örneğin, yalnızca sınırlı miktarda hassas veri toplayan erken aşamadaki bir girişim, ilk güvenlik hedefleri için Seviye 1'e odaklanmaya karar verebilir, ancak bir banka, çevrimiçi bankacılık uygulaması için müşterilerine Seviye 3'ten daha düşük bir seviyeyi haklı çıkarmakta zorlanabilir.

ASVS nasıl kullanılır?

ASVS'nin yapısı

ASVS, toplamda yaklaşık 350 gereksinimden oluşur ve bu gereksinimler 17 bölüme ayrılmıştır. Her bölüm de alt bölümlere ayrılmıştır.

Bölüm ve alt bölümlerin ayrılmasının amacı, uygulama ile ilgili olanları seçmeyi veya filtrelemeyi kolaylaştırmaktır. Örneğin, bir makineler arası API için, V3 bölümündeki web ön uçlarıyla ilgili gereksinimler ilgili olmayacaktır. OAuth veya WebRTC kullanılmıyorsa, bu bölümler de göz ardı edilebilir.

Sürüm stratejisi

ASVS sürümleri "Major.Minor.Patch"("Büyük.Küçük.Yama") modelini izler ve sayılar sürüm içinde nelerin değiştiği hakkında bilgi verir. Büyük bir sürümde ilk sayı, küçük bir sürümde ikinci sayı ve yama sürümünde üçüncü sayı değişir.

- Büyük sürüm - Tamamen yeniden düzenleme yapılmıştır. Gereksinim numaraları da dahil olmak üzere neredeyse her şey değişebilir. Uyumluluk için yeniden değerlendirme gerekli olacaktır (örneğin, 4.0.3 -> 5.0.0).

- Küçük sürüm - Gereksinimler eklenebilir veya kaldırılabilir, ancak genel numaralandırma aynı kalır. Uyumluluk için yeniden değerlendirme gerekli olacaktır, ancak daha kolay olacaktır (örneğin, 5.0.0 -> 5.1.0).
- Yama sürümü - Gereksinimler kaldırılabilir (örneğin, tekrar eden veya güncelliğini yitiren gereksinimler) veya daha az katı hale getirilebilir, ancak önceki sürüme uygun olan bir uygulama, yama sürümüne de uygun olacaktır (örneğin, 5.0.0 -> 5.0.1).

Yukarıdaki bilgiler özellikle ASVS'deki gereksinimlerle ilgilidir. Gereksinimleri çevreleyen metin ve ekler gibi diğer içeriklerdeki değişiklikler, önemli bir değişiklik olarak değerlendirilmeyecektir.

ASVS ile esneklik

Yukarıda açıklanan belgeleme gereksinimleri ve seviye mekanizması gibi bazı noktalar, ASVS'yi daha esnek ve kuruluşa özgü bir şekilde kullanma olanağı sağlar.

Ayrıca, kuruluşların uygulamalarının belirli özelliklerine ve risk seviyelerine göre gereksinimleri ayarlayan, kuruluşa veya alana özgü bir "fork"oluşturmaları şiddetle tavsiye edilir. Ancak, gereksinim 4.1.1'in tüm sürümlerde aynı anlama gelmesi için izlenebilirliği korumak önemlidir.

İdeal olarak, her kuruluş kendi özel ASVS'sini oluşturmalı ve ilgisiz bölümleri (örneğin, kullanılmıyorsa GraphQL, WebSockets, SOAP) çıkarmalıdır. Kuruluşa özgü bir ASVS sürümü veya eki, gereksinimlere uyum sağlarken kullanılacak kütüphaneleri veya kaynakları ayrıntılı olarak açıklayan, kuruluşa özgü uygulama kılavuzları sağlamak için de iyi bir varlıktır.

ASVS Gereksinimlerine Nasıl Atıfta Bulunulur?

Her gereksinim, her ögesinin bir sayı olduğu `<bölüm>.<kısım>.<gereksinim>` biçiminde bir tanımlayıcıya sahiptir. Örneğin, `1.11.3`.

- `<bölüm>` (chapter) değeri, gereksinimin geldiği bölümü belirtir. Örneğin, tüm `1.#.#` gereksinimleri "Kodlama (Encoding) ve Temizleme"bölümünden gelir.
- `<kısım>` (section) değeri, o bölüm içinde gereksinimin yer aldığı kısmı belirtir. Örneğin, tüm `1.2.#` gereksinimleri "Kodlama ve Temizleme"bölümünün "Enjeksiyon Önleme" bölümündedir.
- `<gereksinim>` (requirement) değeri, bölüm ve kısım içindeki belirli bir gereksinimi tanımlar. Örneğin, bu standardın 5.0.0 sürümünde `1.2.5` şu şekildedir:

> Uygulamanın işletim sistemi komut enjeksiyonuna karşı koruma sağladığını ve işletim sistemi çağrılarının parametreli işletim sistemi sorguları veya bağlamsal komut satırı çıktı kodlaması kullandığını doğrulayın.

Tanımlayıcılar standardın sürümleri arasında değişebileceğinden, diğer belgeler, raporlar veya araçlar için aşağıdaki biçimin kullanılması tercih edilir: `v<sürüm>-<bölüm>.<kısım>.<gereksinim>`,

burada: 'sürüm'ASVS sürüm etiketidir. Örneğin: `v5.0.0-1.2.5`, 5.0.0 sürümünün "Kodlama ve Temizleme"bölümünün "Enjeksiyon Önleme"alt bölümündeki 5. gereksinimi ifade eder. (Bu, `v<sürüm>-<gereksinim_tanımlayıcı>` olarak özetlenebilir.)

Not: Format içinde sürüm numarasından önce gelen v her zaman küçük harf olmalıdır.

Tanımlayıcılar v öğesi olmadan kullanıldığında, varsayılan olarak en güncel Uygulama Güvenliği Doğrulama Standardı içeriğine atıfta bulunulduğu kabul edilir. Ancak standart büyüyüp değiştikçe bu durum sorun yaratabilir, bu nedenle yazarlar veya geliştiriciler sürüm öğesini eklemelidir.

ASVS gereksinim listeleri, başvuru veya programatik kullanım için yararlı olabilecek CSV, JSON ve diğer formatlarda sunulmaktadır.

ASVS'yi Çatallama (Forklama)

Kuruluşlar, üç seviyeden birini seçerek veya uygulama risk seviyesine göre gereksinimleri ayarlayan alana özgü bir fork oluşturarak ASVS'den faydalanabilir. Bu tür bir fork, izlenebilirliği koruduğu sürece teşvik edilir, böylece 4.1.1 gereksiniminin karşılanması tüm sürümlerde aynı anlama gelir.

İdeal olarak, her kuruluş kendi özel ASVS'sini oluşturmalı ve alakasız bölümleri (örneğin, kullanılmıyorsa GraphQL, Websockets, SOAP) çıkarmalıdır. Fork oluşturma, ASVS Seviye 1'i temel alarak başlamalı ve uygulamanın riskine göre Seviye 2 veya 3'e ilerlemelidir.

ASVS'nin kullanım alanları

ASVS, bir uygulamanın güvenliğini değerlendirmek amacıyla kullanılabilir ve bu konu bir sonraki bölümde daha ayrıntılı olarak ele alınacaktır. Ancak, ASVS'nin (veya forklanmış bir sürümünün) başka potansiyel kullanım alanları da belirlenmiştir.

Ayrıntılı Güvenlik Mimarisi Rehberi Olarak Kullanma

Uygulama Güvenliği Doğrulama Standardı'nın en yaygın kullanım alanlarından biri, güvenlik mimarları için bir kaynak olarak kullanılmasıdır. Özellikle modern uygulamalarda, güvenli bir uygulama mimarisi oluşturmak için sınırlı sayıda kaynak mevcuttur. ASVS, güvenlik mimarlarının veri koruma modelleri ve girdi doğrulama stratejileri gibi yaygın sorunlar için daha iyi kontroller seçmelerine olanak tanıyarak bu boşlukları doldurmak için kullanılabilir. Mimari ve dokümantasyon gereksinimleri bu konuda özellikle yararlı olacaktır.

Özelleştirilmiş Bir Güvenli Kodlama Referansı Olarak Kullanma

ASVS, uygulama geliştirme sırasında güvenli kodlama referansı hazırlamak için temel olarak kullanılabilir ve geliştiricilerin yazılım oluştururken güvenliği göz önünde bulundurmalarını sağlar. ASVS temel olarak kullanılabilirken, kuruluşlar kendi özel kılavuzlarını hazırlamalıdır. Bu kılavuzlar

açık ve birleşik olmalı ve ideal olarak güvenlik mühendisleri veya güvenlik mimarlarının rehberliğine dayalı olarak hazırlanmalıdır. Bunun bir uzantısı olarak, kuruluşların mümkün olduğunca kılavuzda referans gösterilebilecek ve geliştiriciler tarafından kullanılabilecek onaylanmış güvenlik mekanizmaları ve kütüphaneleri hazırlamaları teşvik edilir.

Otomatik Birim ve Entegrasyon Testleri İçin Rehber Olarak Kullanma

ASVS, yüksek düzeyde test edilebilir şekilde tasarlanmıştır. Bazı doğrulamalar teknik nitelikteyken, bazı gereksinimler (mimari ve dokümantasyon gereksinimleri gibi) dokümantasyon veya mimari incelemesi gerektirebilir. Teknik yollarla doğrulanabilir gereksinimlerle ilgili belirli ve ilgili kötüye kullanım durumlarını test eden ve "fuzz"yapan birim ve entegrasyon testleri oluşturarak, bu kontrollerin her derlemede doğru şekilde çalıştığını kontrol etmek daha kolay hale gelmelidir. Örneğin, bir oturum açma denetleyicisi için test paketi için ek testler oluşturulabilir. Bu testlerde, kullanıcı adı parametresi için yaygın varsayılan kullanıcı adları, hesap numaralandırma, kaba kuvvet saldırısı, LDAP ve SQL enjeksiyonu ve XSS test edilir. Benzer şekilde, şifre parametresi üzerinde yapılan bir testte yaygın şifreler, şifre uzunluğu, null byte enjeksiyonu, parametrenin kaldırılması, XSS ve daha fazlası yer almalıdır.

Güvenli Geliştirme Eğitimi İçin Kullanma

ASVS, güvenli yazılımın özelliklerini tanımlamak için de kullanılabilir. Birçok "güvenli kodlama kursu", kodlama ipuçlarının hafifçe serpiştirildiği etik hackleme kurslarından ibarettir. Bu, geliştiricilerin daha güvenli kod yazmasına yardımcı olmayabilir. Güvenli geliştirme kursları, yapılmaması gereken en önemli 10 olumsuz şey yerine ASVS'yi kullanarak ASVS'de bulunan olumlu mekanizmalara odaklanabilir. ASVS yapısı, bir uygulamayı güvenli hale getirirken farklı konuları ele almak için mantıklı bir yapı da sağlar.

Güvenli Yazılım Satın Alımına Rehberlik Eden Bir Çerçeve Olarak Kullanma

ASVS, güvenli yazılım satın alımına veya özel geliştirme hizmetleri satın alımına yardımcı olan harika bir çerçevedir. Alıcı, satın almak istediği yazılımın ASVS seviye X'te geliştirilmesi gerektiğini belirten bir gereksinim belirleyebilir ve satıcıdan yazılımın ASVS seviye X'i karşıladığını kanıtlamasını isteyebilir.

ASVS'nin Pratikte Uygulanması

Farklı tehditlerin farklı motivasyonları vardır. Bazı sektörler, benzersiz bilgi ve teknoloji varlıklarına ve alana özgü yasal uyumluluk gereksinimlerine sahiptir.

Kuruluşların, işlerinin doğasına göre benzersiz risk özelliklerini derinlemesine incelemeleri ve bu risk ve iş gereksinimlerine göre uygun ASVS seviyesini belirlemeleri şiddetle tavsiye edilir.

Değerlendirme ve Sertifikasyon

OWASP'in ASVS Sertifikaları ve Güven Damgaları Konusundaki Tutumu

OWASP, satıcıdan bağımsız bir kar amacı gütmeyen kuruluş olarak, herhangi bir satıcıyı, doğrulayıcıyı veya yazılımı sertifikalandırmaz. ASVS uyumluluğunu iddia eden herhangi bir güvence, güven damgası veya sertifikasyon OWASP tarafından resmi olarak onaylanmamaktadır. Bu nedenle, kuruluşlar üçüncü tarafların ASVS sertifikasyonu iddialarına karşı dikkatli olmalıdır.

Kuruluşlar, resmi bir OWASP sertifikası sunduklarını iddia etmemek koşuluyla güvence hizmetleri sunabilirler.

ASVS Uyumluluğu Nasıl Doğrulanır?

ASVS, uyumluluğun nasıl doğrulanacağı konusunda kasıtlı olarak kesin bir yöntem belirtmez. Ancak, bazı temel noktaların vurgulanması önemlidir.

Doğrulama Raporlaması

Geleneksel sızma testi raporları, yalnızca başarısız olan durumları istisnalarla listeler. Ancak bir ASVS sertifikasyon raporu; kapsamı, kontrol edilen tüm gereksinimlerin özetini, istisna tespit edilen gereksinimleri ve bu sorunların nasıl çözüleceğine dair rehberliği içermelidir. Bazı gereksinimler uygulanabilir olmayabilir (örneğin, durum bilgisi olmayan API'lerde oturum yönetimi) ve bu durum raporda açıkça belirtilmelidir.

Doğrulama Kapsamı

Uygulama geliştiren bir organizasyon, uygulamanın işlevselliğine bağlı olarak bazı gereksinimleri uygulamayabilir, çünkü bunlar alakasız veya daha az önemli olabilir. Doğrulayıcı kişi veya kurum, doğrulamanın kapsamını açıkça belirtmelidir. Organizasyonun hangi Seviye'yi hedeflediği ve hangi gereksinimlerin dahil edildiği açıklanmalıdır. Bu, nelerin hariç tutulduğundan ziyade, nelerin dahil edildiği bakış açısından ele alınmalıdır. Ayrıca, uygulanmayan gereksinimlerin neden dışlandığına dair gerekçeli bir görüş sunulmalıdır.

Bu yaklaşım, raporu okuyan tarafın doğrulamanın bağlamını anlamasına ve uygulamaya ne kadar güvenebileceğine dair bilinçli bir karar vermesine olanak tanır.

Sertifikasyon yapan kuruluşlar kendi test yöntemlerini seçebilir, ancak bu yöntemler raporda açıkça belirtilmeli ve ideal olarak tekrarlanabilir olmalıdır. Girdi doğrulama gibi konular için sızma testi ya da kaynak kod analizi gibi farklı yöntemler kullanılabilir.

Doğrulama Mekanizmaları

Bazı ASVS gereksinimlerinin doğrulanması için birden fazla teknik gerekebilir. Geçerli kullanıcı bilgileriyle tüm uygulama kapsamı elde edilerek yapılan sızma testlerine ek olarak ASVS gereksinimlerinin doğrulanması; dokümantasyon, kaynak kodu, yapılandırmalar ve geliştirme sürecine dahil kişilere, özellikle Seviye 2 ve Seviye 3 gereksinimlerinin doğrulanması için erişim gerektirebilir. Bulguların belgelenmesi standart bir uygulamadır ve buna çalışma notları, ekran görüntüleri, betikler ve test günlükleri (logs) dahil olabilir. Yalnızca otomatik bir araç çalıştırmak ve derinlemesine test yapmamak sertifikasyon için yeterli değildir; her gereksinimin doğrulanabilir şekilde test edilmiş olması gerekir.

ASVS gereksinimlerinin otomasyonla doğrulanması, her zaman ilgi çeken bir konudur. Bu nedenle, otomatik ve kara kutu (black box) testlerle ilgili bazı noktaların netleştirilmesi önemlidir.

Otomatik Güvenlik Testi Araçlarının Rolü

DAST ve SAST gibi dinamik ve statik güvenlik test araçları doğru şekilde build pipeline'a entegre edildiğinde, asla var olmaması gereken bazı güvenlik açıklarını tespit edebilir. Ancak, dikkatli bir şekilde yapılandırılmaz ve ayarlanmazlarsa, yeterli kapsama alanı sağlamazlar ve yüksek sayıda yanlış pozitif, gerçek güvenlik sorunlarının tespitini ve düzeltilmesini engeller.

Bu araçlar, çıktı kodlaması veya veri temizleme gibi bazı temel teknik gereksinimlerin doğrulanmasında yardımcı olabilir. Ancak, iş mantığı veya erişim kontrolü gibi daha karmaşık ASVS gereksinimlerinin doğrulanmasında yetersiz kalacaklardır.

Daha karmaşık gereksinimler için otomasyon yine de kullanılabilir, ancak uygulamaya özel testlerin yazılması gerekebilir. Bunlar, organizasyonun halihazırda kullandığı birim ve entegrasyon testlerine benzer şekilde tasarlanabilir. Mevcut test altyapısı kullanılarak bu ASVS'ye özel testler yazılabilir. Bu kısa vadede yatırım gerektirse de, uzun vadede bu gereksinimlerin sürekli olarak doğrulanabilmesi açısından önemli bir fayda sağlar.

Özetle, "otomasyon ile test edilebilir"ifadesi "hazır bir aracı çalıştırmak"anlamına gelmez.

Sızma Testlerinin Rolü

4.0 sürümünde Seviye 1, kara kutu (black box) test için optimize edilmişti (yani dokümantasyon veya kaynak kod olmadan). Ancak o zaman bile bunun etkili bir güvence yöntemi olmadığı ve aktif olarak kaçınılması gerektiği vurgulanmıştır.

Gerekli ek bilgilere erişim olmadan test yapmak, güvenlik doğrulaması açısından hem verimsiz hem de etkisizdir. Bu durum, kaynak kodun incelenmesi, tehditlerin ve eksik kontrollerin tespiti gibi daha kapsamlı analizlerin yapılmasını engeller ve süreci uzatır.

Geleneksel sızma testlerinin yerini, dokümantasyon ve/veya kaynak kod temelli (hibrit) sızma testlerinin almasını şiddetle tavsiye ediyoruz. Geliştirici ekip ve uygulamanın dokümantasyonuna tam erişim ile yapılan bu testler, birçok ASVS gereksiniminin doğrulanması için gerekli olacaktır.

v4.x ile Karşılaştırıldığında Yapılan Değişiklikler

Giriş

Standartın 4.x sürümüne aşina olan kullanıcılar, 5.0 sürümünde yapılan temel değişiklikleri içerik, kapsam ve altında yatan felsefe bakımından gözden geçirmeyi faydalı bulabilirler.

Sürüm 4.0.3'te yer alan 286 gereksinimin yalnızca 11 tanesi hiçbir değişikliğe uğramadan korunmuştur. 15 gereksinimde ise yalnızca anlamı değiştirmeyen dilbilgisel düzeltmeler yapılmıştır. Toplamda 109 gereksinim (%38) artık 5.0 sürümünde bağımsız birer gereksinim olarak yer almamaktadır. Bunların 50'si doğrudan silinmiş, 28'i yinelenen olarak kaldırılmış ve 31'i başka gereksinimlerle birleştirilmiştir. Geri kalan tüm gereksinimler bir şekilde revize edilmiştir. İçeriği esasen değiştirilmemiş olan gereksinimler bile, yeniden sıralama veya yapısal değişiklikler nedeniyle artık farklı tanımlayıcılara (ID) sahiptir.

Sürüm 5.0'ın benimsenmesini kolaylaştırmak için, kullanıcıların sürüm 4.x'teki gereksinimlerin sürüm 5.0'daki gereksinimlerle nasıl eşleştiğini izlemelerine yardımcı olmak için eşleme belgeleri sağlanmıştır. Bu eşlemeler sürüm numaralandırmasına bağlı değildir ve gerektiğinde güncellenebilir veya açıklığa kavuşturulabilir.

Gereksinim Felsefesi

Kapsam ve Odak

Sürüm 4.x, standardın hedeflenen kapsamıyla uyumlu olmayan bazı gereksinimler içeriyordu. Bu gereksinimler kaldırılmıştır. 5.0 sürümünün kapsam kriterlerini karşılamayan veya doğrulanabilir olmayan gereksinimler de eklenen gereksinimlerden hariç tutulmuştur.

Mekanizmalar Yerine Güvenlik Hedeflerine Önem Verilmesi

Sürüm 4.x'te birçok gereksinim, altında yatan güvenlik hedefleri yerine belirli mekanizmalara odaklanıyordu. 5.0 sürümünde ise gereksinimler, belirli güvenlik hedefleri etrafında yapılandırılmıştır. Belirli mekanizmalara yalnızca bunların tek pratik çözüm olması durumunda doğrudan referans verilmiş veya örnek/rehber olarak yer verilmiştir.

Bu yaklaşım, belirli bir güvenlik hedefinin birden fazla yolla gerçekleştirilebileceğini kabul eder ve kuruluşların esnekliğini kısıtlayabilecek gereksiz derecede kuralcı yönlendirmelerden kaçınır.

Ayrıca, aynı güvenlik sorununu ele alan gereksinimler uygun şekilde birleştirilmiştir.

Belgelenmiş Güvenlik Kararları

"Belgelenmiş güvenlik kararları"kavramı 5.0 sürümünde yeni gibi görünse de, 4.0 sürümündeki politika uygulamaları ve tehdit modellemesi ile ilgili daha önceki gereksinimlerin evrimleşmiş halidir.

Önceki sürümlerde bazı gereksinimler, örneğin izin verilen ağ bağlantılarının belirlenmesi gibi, güvenlik kontrollerinin uygulanmasına yön verecek analizlerin yapılmasını dolaylı olarak talep etmekteydi.

Gerekli bilgilerin hem uygulama hem de doğrulama süreçlerinde erişilebilir olmasını sağlamak amacıyla bu beklentiler artık açıkça belgelenmesi gereken gereksinimler olarak tanımlanmıştır. Böylece bu gereksinimler daha net, uygulanabilir ve doğrulanabilir hale getirilmiştir.

Yapısal Değişiklikler ve Yeni Bölümler

Sürüm 5.0'daki bazı bölümler tamamen yeni içerikler sunmaktadır:

- OAuth ve OIDC –Bu protokollerin erişim yetkisi devri ve tek oturum açma (SSO) için yaygın biçimde benimsenmiş olması nedeniyle, geliştiricilerin karşılaşabileceği çeşitli senaryoları kapsayan özel gereksinimler eklenmiştir. Bu alan, geçmişte Mobil ve IoT gereksinimlerinin ayrı bir standarda dönüştürülmesinde olduğu gibi, zamanla bağımsız bir standarda da dönüşebilir.
- WebRTC –Bu teknolojinin yaygınlaşmasıyla birlikte, kendine özgü güvenlik gereksinimleri ve zorlukları artık özel bir bölümde ele alınmaktadır.

Bölüm ve kısımların birbiriyle ilişkili gereksinim kümeleri etrafında daha tutarlı şekilde düzenlenmesine de özen gösterilmiştir.

Bu yeniden yapılandırma kapsamında şu ek bölümler oluşturulmuştur:

- Kendi İçinde Taşıyıcı Token'lar –Daha önce oturum yönetimi altında yer alan bu mekanizmalar artık ayrı bir yapı olarak tanınmakta ve OAuth ile OIDC gibi durumsuz iletişim yapılarının temel unsurlarından biri olarak ele alınmaktadır. Kendilerine özgü güvenlik etkileri nedeniyle, 5.x sürümünde bazı yeni gereksinimlerle birlikte ayrı bir bölümde toplanmıştır.
- Web Ön Yüz Güvenliği (Frontend Security) –Tarayıcı tabanlı uygulamaların artan karmaşıklığı ve yalnızca API'lara dayalı mimarilerin yaygınlaşması nedeniyle, frontend güvenliği gereksinimleri ayrı bir bölümde ele alınmıştır.
- Güvenli Kodlama ve Mimari –Diğer mevcut bölümlere doğrudan uymayan genel güvenlik uygulamalarıyla ilgili yeni gereksinimler burada gruplandırılmıştır.

Sürüm 5.0'daki diğer düzenleyici değişiklikler ise niyetin daha net şekilde anlaşılmasını sağlamayı amaçlamıştır. Örneğin, girdi doğrulama gereksinimleri iş kuralları doğrultusunda işletim sağladıkları için artık iş mantığı ile birlikte gruplanmış, eski yapıda yer aldıkları temizleme ve kodlama (sanitization & encoding) konularından ayrılmıştır.

Önceki V1 Mimari bölümü tamamen kaldırılmıştır. Bu bölümün ilk kısmı kapsam dışı gereksinimler içeriyordu. Sonraki kısımlar ise ilgili yeni bölümlere taşınmış, gereksinimler sadeleştirilmiş ve gerektiğinde yinelenen maddeler birleştirilerek açıklığa kavuşturulmuştur.

Diğer Standartlara Yapılan Doğrudan Eşlemelerin Kaldırılması

Standardın ana kısmında yer alan diğer standartlara yönelik doğrudan eşlemeler kaldırılmıştır. Bunun yerine, ASVS'nin OWASP projeleri ve harici standartlarla ilişkilendirilmesini sağlayacak OWASP Common Requirement Enumeration (CRE) projesiyle bir eşleme hazırlanması hedeflenmektedir.

Aşağıda açıklanan nedenlerle, CWE ve NIST gibi standartlara yapılan doğrudan eşlemeler artık devam ettirilmemektedir.

NIST Dijital Kimlik Kılavuzları (Digital Identity Guidelines) ile Azaltılmış Bağlılık

NIST Digital Identity Guidelines (SP 800-63) uzun süredir kimlik doğrulama ve yetkilendirme kontrolleri için bir başvuru noktası olarak kullanılmıştır. Sürüm 4.x'te bazı bölümler, NIST'in yapısı ve terminolojisiyle oldukça örtüşüyordu.

Bu kılavuzlar hâlâ önemli bir referans noktası olsa da, böyle bir sıkı örtüşme birtakım sorunlara yol açmıştır. Bu sorunlara yaygın olarak benimsenmemiş terimlerin kullanımı, benzer gereksinimlerin yinelenmesi ve eşlemenin eksik kalması örnek verilebilir. Sürüm 5.0, bu yaklaşımdan uzaklaşarak daha açık ve uygulanabilir bir yapı sunmayı hedeflemektedir.

Common Weakness Enumeration'dan (CWE) Uzaklaşma

Common Weakness Enumeration (CWE), yazılım güvenliği zafiyetleri için faydalı bir sınıflandırma sistemidir. Ancak, yalnızca kategori düzeyinde tanımlanmış CWE'ler, bir gereksinimin yalnızca tek bir CWE ile eşleştirilememesi gibi eşleme zorlukları ve sürüm 4.x'teki bazı belirsiz eşlemeler sürüm 5.0'da doğrudan CWE eşlemelerinin sonlandırılmasına neden olmuştur.

Seviye Tanımlarının Yeniden Düşünülmesi

Sürüm 4.x'te seviyeler L1 ("Minimum"), L2 ("Standart") ve L3 ("Gelişmiş") olarak tanımlanıyordu. Hassas veri işleyen tüm uygulamaların en az L2 seviyesini sağlaması gerektiği ima ediliyordu.

Sürüm 5.0, bu yaklaşıma dair çeşitli sorunları ele almaktadır.

Sürüm 4.x'te seviye göstergeleri tik işaretleriyle ifade edilirken, 5.x sürümünde tüm formatlarda (Markdown, PDF, DOCX, CSV, JSON ve XML) basit sayılar kullanılmaktadır. Geriye dönük uyumluluk için eski CSV, JSON ve XML çıktı formatlarında onay işaretleri hâlâ kullanılmaktadır.

Daha Kolay Giriş Seviyesi

Geri bildirimler, yaklaşık 120 maddeden oluşan Seviye 1 gereksinimlerinin, aynı zamanda "minimum"olarak etiketlenip birçok uygulama için yetersiz görülmesi nedeniyle benimsenmesinin zorlaştığını göstermiştir. Sürüm 5.0, bu eşiği düşürmeyi hedefleyerek Seviye 1'i öncelikli olarak ilk

savunma katmanına ait gereksinimler etrafında tanımlar ve bu seviyedeki gereksinimleri daha az ve daha net hale getirir. Sayısal olarak örnek vermek gerekirse sürüm 4.0.3'te 278 gereksinimden 128'i Seviye 1'di (%46). Sürüm 5.0.0'da ise 345 gereksinimden yalnızca 70'i Seviye 1'dir (%20).

Test Edilebilirlik Yanılgısı

Sürüm 4.x'te Seviye 1 gereksinimleri belirlenirken, "dışsal kara kutu sızma testleriyle değerlendirilebilirlik"ön plandaydı. Ancak bu yaklaşım, Seviye 1'in güvenlik kontrolleri için minimum bir temel olması yönündeki amacıyla tam olarak örtüşmüyordu. Bazı kullanıcılar, Seviye 1'in yetersiz olduğunu savunurken, bazıları ise uygulanabilirliğini zor buluyordu.

Test edilebilirliğe odaklanmak hem göreceli hem de zaman zaman yanıltıcı olabilir. Bir gereksinimin test edilebilir olması, onun kolayca veya otomatik biçimde test edilebileceği anlamına gelmez. Ayrıca, en kolay test edilebilen gereksinimler her zaman en yüksek güvenlik katkısına sahip ya da en kolay uygulanabilir olanlar değildir.

Bu nedenle sürüm 5.0'da seviye belirleme kararları, esas olarak risk azaltımı temeline dayanır ve uygulama çabası da göz önünde bulundurulmuştur.

Sadece Risk Odaklı Değil

Belirli uygulamalar için belirli bir seviyeyi zorunlu kılan, katı ve risk bazlı seviye yaklaşımı fazlasıyla sert bir yapıya dönüşmüştür. Gerçekte, güvenlik kontrollerinin önceliklendirilmesi ve uygulanması yalnızca risk değil, aynı zamanda uygulama çabası, organizasyonel olgunluk ve kullanıcıya verilmek istenen güvenlik mesajı gibi çeşitli faktörlere bağlıdır.

Bu nedenle, kuruluşların olgunluk düzeylerine ve kullanıcılarına vermek istedikleri mesaja göre hangi seviyeyi hedeflemeleri gerektiğine kendilerinin karar vermesi teşvik edilmektedir.

V1 Kodlama (Encoding) ve Temizleme

Kontrol Amacı

Bu bölüm, güvenilmeyen verilerin güvensiz işlenmesiyle ilişkili en yaygın web uygulaması güvenlik zayıflıklarına odaklanır. Bu tür durumlar, güvenilmeyen verinin ilgili yorumlayıcının sözdizimi kuralları çerçevesinde yanlış şekilde yorumlanmasına neden olan çeşitli teknik güvenlik açıklarına yol açar.

Modern web uygulamalarında, parametreli sorgular, otomatik kaçış (escaping) veya şablonlama (templating) çerçeveleri gibi daha güvenli API'ların kullanımı her zaman en iyi yaklaşımdır. Aksi takdirde ya da bu mümkün değilse, çıktı kodlaması, kaçış veya temizleme işlemlerinin dikkatle uygulanması uygulamanın güvenliği için kritik öneme sahiptir.

Girdi doğrulama, beklenmeyen ve tehlikeli içeriklere karşı savunma derinliği mekanizması olarak kullanılabilir. Ancak asıl amacı gelen verinin işlevsel ve iş gereksinimleriyle uyumlu olup olmadığını kontrol etmektir. Bu nedenle, bu konudaki gereksinimler "Doğrulama ve İş Mantığı"bölümünde ele alınmıştır.

V1.1 Kodlama ve Temizleme Mimarisi

Aşağıdaki bölümlerde, güvenli olmayan içeriklerin güvenli şekilde işlenmesine yönelik, sözdizimi veya yorumlayıcıya özel güvenlik gereksinimleri sunulmaktadır. Bu bölümdeki gereksinimler, bu işlemlerin hangi sırayla ve nerede gerçekleştirilmesi gerektiğini kapsar. Ayrıca, verilerin saklandığı her durumda orijinal (ham) haliyle saklanması gerektiğini (örneğin HTML kodlaması gibi kodlanmış/kaçırılmış biçimde değil) vurgular. Bu sayede çift kodlama (double encoding) gibi sorunlar önlenmiş olur.

#	Açıklama	Seviye
1.1.1	Girdinin yalnızca bir kez kanonik forma (canonical form) çözümlendiği veya kaçış işlemlerinin kaldırıldığı, bu işlemin yalnızca verinin bu biçimde beklenmesi durumunda yapıldığı ve bu işlemin girdi doğrulama veya temizleme işlemlerinden sonra değil, önce gerçekleştiği doğrulanmalıdır.	2
1.1.2	Uygulamanın, çıktıyı kullanılması gereken yorumlayıcıya verilmeden hemen önce veya yorumlayıcının kendisi tarafından çıktı kodlaması ya da kaçış işlemi yaptığı doğrulanmalıdır.	2

V1.2 Enjeksiyon Önleme

Potansiyel olarak tehlikeli içeriklerin yakınında yapılan çıktı kodlama veya kaçış işlemleri, uygulama güvenliği için kritiktir. Genellikle çıktı kodlama ve kaçış işlemleri kalıcı değildir. Yalnızca verinin anlık olarak yorumlayıcı tarafından güvenli şekilde işlenebilmesi için uygulanır. Bu işlemin çok erken yapılması, içeriğin bozulmasına veya etkisiz hale gelmesine neden olabilir.

Çoğu durumda, yazılım kütüphaneleri bu işlemleri otomatik olarak gerçekleştiren güvenli veya daha güvenli işlevler sağlar. Ancak, bu işlevlerin geçerli bağlama uygun olduğundan emin olunmalıdır.

#	Açıklama	Seviye
1.2.1	HTTP yanıtı, HTML veya XML belgesi için yapılan çıktı kodlamasının; HTML elemanları, HTML öznitelikleri, HTML yorumları, CSS ya da HTTP başlık alanları gibi bağlamlara uygun olduğu doğrulanmalıdır. Böylece mesaj veya belge yapısının bozulması önlenir.	1

#	Açıklama	Seviye
1.2.2	Dinamik olarak URL oluşturulurken, güvenilmeyen verinin bağlamına göre (örneğin, sorgu veya yol parametreleri için URL kodlaması ya da base64url kodlaması) kodlandığı doğrulanmalıdır. Yalnızca güvenli URL protokollerine izin verilmelidir (örneğin, javascript: veya data: yasaklanmalıdır).	1
1.2.3	JavaScript içeriği (JSON dahil) dinamik olarak oluşturulurken çıktı kodlama/kaçış kullanıldığı doğrulanmalıdır. Böylece mesaj veya belge yapısı değiştirilmemiş olur (JavaScript ve JSON enjeksiyonlarının önüne geçilir).	1
1.2.4	Veri sorguları veya veritabanı sorguları (SQL, HQL, NoSQL, Cypher vb.) için parametreli sorgular, ORM'ler, entity framework'ler kullanıldığı ya da SQL enjeksiyonu ve diğer veritabanı enjeksiyon saldırılarına karşı koruma sağlandığı doğrulanmalıdır. Bu durum, saklı yordam (stored procedure) yazarken de geçerlidir.	1
1.2.5	Uygulamanın işletim sistemi komut enjeksiyonlarına karşı korunduğu ve işletim sistemi çağrılarında parametreli OS sorguları veya bağlama özgü komut satırı çıktı kodlamasının kullanıldığı doğrulanmalıdır.	1
1.2.6	Uygulamanın LDAP enjeksiyon zafiyetlerine karşı korunduğu ya da LDAP enjeksiyonlarını önlemek için özel güvenlik kontrollerinin uygulandığı doğrulanmalıdır.	2
1.2.7	XPath enjeksiyon saldırılarına karşı, sorgu parametrelemesi veya önceden derlenmiş sorgular kullanılarak koruma sağlandığı doğrulanmalıdır.	2
1.2.8	LaTeX işlemcilerinin güvenli şekilde yapılandırıldığı (örneğin, –shell-escape bayrağının kullanılmadığı) ve yalnızca izin verilen komutlardan oluşan bir listeyle çalıştırıldığı doğrulanmalıdır.	2
1.2.9	Düzenli ifadelerde (regex) özel karakterlerin (genellikle ters eğik çizgi ile) kaçırılarak meta karakter olarak yorumlanmalarının önlendiği doğrulanmalıdır.	2
1.2.10	Uygulamanın CSV ve formül enjeksiyonlarına karşı korunduğu doğrulanmalıdır. Uygulama, CSV içeriği dışa aktarırken RFC 4180 Bölüm 2.6 ve 2.7'de tanımlanan kaçış kurallarına uymalıdır. Ayrıca, CSV veya diğer tablo formatlarına (XLS, XLSX, ODF gibi) aktarım yapılırken, alanın ilk karakteri olarak çıkan özel karakterler ('=', '+', '-', '@', '\t'(sekme), '\0'(null karakter)) tek tırnak (') ile kaçırılmalıdır.	3

Not: Parametreli sorguların veya SQL kaçışlarının kullanılması her zaman yeterli değildir. Örneğin, tablo ve sütun adları (özellikle "ORDER BY"gibi) kaçırılamaz. Bu alanlarda kullanıcı tarafından sağlanan veriler kaçırılmış olarak dahil edilirse sorgular başarısız olur veya SQL enjeksiyonuna

neden olabilir.

V1.3 Temizleme (Sanitization)

Güvenilmeyen içeriğin güvensiz bir bağlamda kullanılmasına karşı ideal koruma yöntemi, önceki bölümde detaylı olarak açıklandığı gibi, bağlama özel kodlama veya kaçıştırma (escaping) kullanmaktır. Bu yöntem, içeriğin anlamını koruyarak güvenli hale getirir.

Bu mümkün olmadığında, potansiyel olarak tehlikeli karakterlerin veya içeriğin temizlenmesi (sanitize) gerekir. Bazı durumlarda, bu işlem girdinin anlamını değiştirebilir; ancak güvenlik açısından başka bir seçenek olmayabilir.

#	Açıklama	Seviye
1.3.1	WYSIWYG editörlerinden veya benzer kaynaklardan gelen tüm güvenilmeyen HTML girdilerinin, bilinen ve güvenli bir HTML temizleme kütüphanesi ya da çerçeve özelliği ile temizlendiği doğrulanmalıdır.	1
1.3.2	Uygulamanın eval() veya Spring Expression Language (SpEL) gibi dinamik kod çalıştırma özelliklerini kullanmaktan kaçındığı doğrulanmalıdır. Alternatif yoksa, kullanıcı girdisi çalıştırılmadan önce mutlaka temizlenmelidir.	1
1.3.3	Potansiyel olarak tehlikeli bir bağlamda kullanılacak verilerin, bu bağlama uygun karakterlere izin vermek ve çok uzun girdileri kısaltmak gibi güvenlik önlemleriyle önceden temizlendiği doğrulanmalıdır.	2
1.3.4	Kullanıcı tarafından sağlanan SVG içeriğinin, yalnızca güvenli etiket ve özellikler (örneğin grafik çizme ile ilgili) içerdiğinden emin olmak için onaylandığı veya temizlendiği doğrulanmalıdır (ör. script veya foreignObject içermemelidir).	2
1.3.5	Markdown, CSS, XSL stil sayfaları, BBCode veya benzeri ifade şablon dillerinde kullanıcı girdisinin temizlendiği veya bu tür özelliklerin devre dışı bırakıldığı doğrulanmalıdır.	2
1.3.6	Uygulamanın, güvenilmeyen verileri başka servislere iletmeden önce izinli protokol, alan adı, yol ve port listelerine göre doğruladığı ve tehlikeli karakterleri temizlediği doğrulanarak SSRF (Sunucu Taraflı İstek Sahteciliği) saldırılarına karşı korunduğu doğrulanmalıdır.	2
1.3.7	Uygulamanın şablon enjeksiyon saldırılarına karşı, güvenilmeyen girdilere dayalı şablon oluşturmayı engellediği; eğer bu kaçınılmazsa, bu girdilerin mutlaka temizlendiği veya sıkı şekilde onaylandığı doğrulanmalıdır.	2

#	Açıklama	Seviye
1.3.8	JNDI sorgularında kullanılmadan önce güvenilmeyen girdilerin temizlendiği ve JNDI'nin enjeksiyonlara karşı güvenli şekilde yapılandırıldığı doğrulanmalıdır.	2
1.3.9	Memcache'e gönderilmeden önce içeriğin temizlenerek enjeksiyon saldırılarına karşı korunduğu doğrulanmalıdır.	2
1.3.10	Format dizgilerinin (format strings) beklenmeyen veya zararlı şekilde çözümlenmesini önlemek için işlenmeden önce temizlendiği doğrulanmalıdır.	2
1.3.11	SMTP veya IMAP enjeksiyonlarını önlemek için, kullanıcı girdisinin posta sistemlerine iletilmeden önce temizlendiği doğrulanmalıdır.	2
1.3.12	Regex girdilerinin üstel geri izlemeye (exponential backtracking) neden olan öğeler içermediği ve güvenilmeyen girdilerin ReDoS (Regex Denial of Service) gibi saldırılara karşı temizlendiği doğrulanmalıdır.	3

V1.4 Bellek, Dizgi ve Yönetilmeyen Kod

Aşağıdaki gereksinimler, sistem programlama dilleri veya yönetilmeyen kod kullanan uygulamalarda görülen güvensiz bellek kullanımıyla ilişkili riskleri ele alır.

Bazı durumlarda, taşma korumaları ve uyarılar, yığın (stack) rastgeleleştirme ve veri yürütme engelleme gibi önlemleri etkinleştiren derleyici bayrakları (compiler flags) kullanılarak bu risklerin önüne geçilebilir. Ayrıca, tehlikeli işaretçi, bellek, format dizgisi, tamsayı ve dizgi işlemleri tespit edildiğinde derlemenin başarısız olmasını sağlayan ayarlar tercih edilmelidir.

#	Açıklama	Seviye
1.4.1	Yığın, arabellek (buffer) veya yığın belleği taşmalarını önlemek ya da tespit etmek amacıyla, bellek güvenli dizgi işlemlerinin, daha güvenli bellek kopyalama ve işaretçi aritmetiğinin kullanıldığı doğrulanmalıdır.	2
1.4.2	Tamsayı taşmalarını önlemek için işaret (sign), aralık (range) ve giriş doğrulama tekniklerinin kullanıldığı doğrulanmalıdır.	2
1.4.3	Dinamik olarak ayrılan belleğin ve kaynakların serbest bırakıldığı ve boşaltılan belleklere ait referansların veya işaretçilerin kaldırıldığı veya null yapıldığı doğrulanmalıdır (dangling pointer ve use-after-free zafiyetlerine karşı).	2

V1.5 Güvenli Ters Serileştirme (Safe Deserialization)

Verilerin saklanmış veya iletilmiş bir temsilinden uygulama nesnelerine dönüştürülmesi işlemi olan ters serileştirme, geçmişte çeşitli kod enjeksiyonu zafiyetlerinin kaynağı olmuştur. Bu işlemin dikkatli ve güvenli bir şekilde yapılması, bu tür güvenlik açıklarının önlenmesi açısından son derece önemlidir.

Özellikle, bazı ters serileştirme yöntemleri, kullanılan programlama dili veya çerçevenin dokümantasyonunda güvensiz olarak tanımlanmıştır ve güvenilmeyen verilerle kullanıldığında güvenli hâle getirilemez. Kullanılan her yöntem için dikkatli bir inceleme yapılmalıdır.

#	Açıklama	Seviye
1.5.1	XML ayrıştırıcılarının (parser) kısıtlayıcı şekilde yapılandırıldığı ve dış varlık çözümleme (external entity resolution) gibi güvensiz özelliklerin XXE (XML eXternal Entity) saldırılarına karşı devre dışı bırakıldığı doğrulanmalıdır .	1
1.5.2	Güvenilmeyen verilerin ters serileştirilmesinde, nesne türleri için izinli liste kullanımı veya istemci tanımlı nesne türlerinin kısıtlanması gibi güvenli giriş işleme önlemlerinin uygulandığı doğrulanmalıdır. Güvensiz olarak tanımlanmış ters serileştirme mekanizmaları, güvenilmeyen girdilerle kullanılmamalıdır.	2
1.5.3	Aynı veri türü için uygulamada kullanılan farklı parser'ların (ör. JSON, XML, URL parser'ları), veriyi tutarlı şekilde ayrıştırdığı ve aynı karakter kodlamasını kullandığı doğrulanmalıdır. Böylece JSON birlikte çalışabilirlik açıkları, URI veya dosya ayrıştırma farkları gibi sorunlar nedeniyle RFI veya SSRF saldırılarının istismar edilmesi önlenir.	3

Referanslar

Daha fazla bilgi için:

- OWASP LDAP Injection Prevention Cheat Sheet
- OWASP Cross Site Scripting Prevention Cheat Sheet
- OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet
- OWASP XML External Entity Prevention Cheat Sheet
- OWASP Testing Guide 4.0: Client-Side Testing
- OWASP Java Encoding Project
- DOMPurify - Client-side HTML Sanitization Library
- RFC4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files

Ters serileştirme ve parser konularında daha fazla bilgi için:

- OWASP Deserialization Cheat Sheet

- OWASP Deserialization of Untrusted Data Guide
- An Exploration of JSON Interoperability Vulnerabilities
- Orange Tsai - A New Era of SSRF Exploiting URL Parser In Trending Programming Languages

V2 Doğrulama ve İş Mantığı

Kontrol Amacı

Bu bölümün amacı, doğrulanan bir uygulamanın aşağıdaki üst düzey hedefleri karşıladığından emin olmaktır:

- Uygulamanın aldığı girdiler, işlevsel veya iş gereksinimleriyle uyumludur.
- İş mantığı akışı sıralıdır, sırayla işlenir ve atlanamaz.
- İş mantığı, sürekli küçük para transferleri veya her seferinde 1 tane olmak üzere 1 milyon arkadaş ekleme gibi otomatik saldırıları tespit edip engellemek için sınırlar ve kontroller içerir.
- Yüksek değerli iş mantığı akışları, kötüye kullanım senaryoları ve kötü niyetli aktörler göz önüne alınarak oluşturulmuştur ve kimlik sahteciliği, manipülasyon, bilgi sızıntısı ve yetki yükseltme saldırılarına karşı korumaya sahiptir.

V2.1 Doğrulama ve İş Mantığı Dokümantasyonu

Doğrulama ve iş mantığı dokümantasyonu; iş mantığı sınırlarını, doğrulama kurallarını ve birleştirilmiş veri öğelerinin bağlamsal tutarlılığını açıkça tanımlamalıdır. Böylece uygulamada neyin uygulanması gerektiği net olur.

#	Açıklama	Seviye
2.1.1	Uygulama dokümantasyonunda, veri öğelerinin beklenen yapıya göre nasıl doğrulanacağına ilişkin giriş doğrulama kurallarının tanımlandığı doğrulanmalıdır. Bu; kredi kartı numaraları, e-posta adresleri, telefon numaraları gibi yaygın veri biçimleri olabileceği gibi iç sistemlere özel biçimler de olabilir.	1
2.1.2	Uygulama dokümantasyonunda, birleştirilmiş veri öğelerinin mantıksal ve bağlamsal tutarlılığının nasıl doğrulanacağı tanımlanmalıdır (örneğin, mahalle ve posta kodunun eşleşip eşleşmediğinin kontrolü).	2
2.1.3	Hem kullanıcı bazında hem de uygulama genelinde geçerli olacak şekilde, iş mantığı sınırları ve doğrulama beklentilerinin belgelendiği doğrulanmalıdır.	2

V2.2 Girdi Doğrulama

Etkili girdi doğrulama kontrolleri, uygulamanın beklediği veri türüne yönelik işlevsel ya da iş gereksinimlerinin sağlanmasını zorunlu kılar. Bu sayede veri kalitesi artar ve saldırı yüzeyi daraltılır. Ancak bu, verinin başka bileşenlerde kullanılmadan veya çıktıya aktarılmadan önce doğru şekilde kodlanması, parametreleştirilmesi ya da temizlenmesi ihtiyacını ortadan kaldırmaz.

Bu bağlamda "girdi"; HTML form alanları, REST istekleri, URL parametreleri, HTTP başlıkları, çerezler, disk dosyaları, veritabanları veya harici API'lerden gelebilir.

Bir iş mantığı kontrolü, belirli bir girdinin 100'den küçük bir sayı olup olmadığını kontrol edebilir. Bir işlevsel beklenti ise bu sayının belirli bir eşiğin altında olmasını isteyebilir, çünkü bu sayı örneğin bir döngünün kaç kez çalışacağını belirliyor olabilir; büyük bir değer, aşırı işlemeye ve hizmet reddi (DoS) durumuna yol açabilir.

Şema doğrulaması zorunlu tutulmamış olsa da, HTTP API'leri veya JSON/XML kullanan diğer arayüzlerin tamamen doğrulanması için en etkili yöntem olabilir.

Şema doğrulamasıyla ilgili aşağıdaki hususlara dikkat edilmelidir:

- JSON Schema doğrulama standardının "yayınlanmış sürümü"üretime hazır kabul edilmekte ancak "stabil"değildir. Kullanıldığında, aşağıdaki gereksinimlerle çelişmediğinden emin olunmalıdır.
- Kullanılan JSON Schema kütüphaneleri, standart resmileştiğinde izlenmeli ve gerekirse güncellenmelidir.
- DTD'lere karşı yapılan XXE saldırılarını önlemek için DTD doğrulaması kullanılmamalı ve çerçeve içindeki DTD işleme özellikleri devre dışı bırakılmalıdır.

#	Açıklama	Seviye
2.2.1	Girdilerin, işlevsel ya da iş gereksinimlerine uygunluğunun doğrulandığı teyit edilmelidir. Bu işlem; izinli değerler, desenler ve aralıklar listesine karşı pozitif doğrulama ile ya da girdinin beklenen yapıya ve mantıksal sınırlara uygunluğunun önceden tanımlanmış kurallara göre karşılaştırılması ile yapılmalıdır. Seviye 1 için bu, güvenlik veya iş kararlarında kullanılan girdilere odaklanabilir. Seviye 2 ve üstü için tüm girdilere uygulanmalıdır.	1
2.2.2	Uygulamanın, güvenilir bir servis katmanında girdi doğrulama zorunluluğu uygulayacak şekilde tasarlandığı doğrulanmalıdır. İstemci tarafında gerçekleştirilen doğrulama, kullanılabilirliği artırabilir ve teşvik edilmelidir. Ancak bu doğrulama,güvenlik kontrolü olarak kabul edilmemelidir.	1
2.2.3	Uygulamanın, ilişkili veri öğelerinin önceden tanımlanmış kurallara göre makul olup olmadığını kontrol ettiği doğrulanmalıdır.	2

V2.3 İş Mantığı Güvenliği

Bu bölüm, uygulamanın iş mantığı süreçlerini doğru şekilde yürüttüğünden ve bu mantık ve akışın istismarına karşı savunmasız olmadığından emin olunması için temel gereksinimleri kapsar.

#	Açıklama	Seviye
2.3.1	Uygulamanın, aynı kullanıcı için iş mantığı akışlarını yalnızca beklenen adım sırasına göre ve adım atlamadan işlediği doğrulanmalıdır.	1
2.3.2	İş mantığı sınırlarının, uygulama dokümantasyonuna uygun şekilde uygulandığı doğrulanmalıdır.	2
2.3.3	İş mantığı seviyesinde işlemlerin (transaction) kullanıldığı ve bir iş mantığı işleminin ya bütünüyle başarıyla tamamlandığı ya da önceki geçerli duruma geri döndürüldüğü (rollback) doğrulanmalıdır.	2
2.3.4	İş mantığı düzeyinde kilitleme mekanizmalarının, sınırlı kaynakların (ör. sinema koltukları, teslimat zamanları) uygulamanın mantığı manipüle edilerek çift rezervasyona neden olmamasını sağladığı doğrulanmalıdır.	2
2.3.5	Yüksek değerli iş mantığı akışlarının, yetkisiz veya kazara gerçekleşebilecek işlemleri önlemek için çoklu kullanıcı onayı gerektirdiği doğrulanmalıdır. Bu işlemler büyük para transferleri, sözleşme onayları, gizli bilgilere erişim veya üretimde güvenlik kurallarının geçersiz kılınmaları gibi durumları içerebilir.	3

V2.4 Otomasyon Karşıtı Önlemler

Bu bölümde, insan benzeri etkileşimlerin zorunlu kılındığı ve aşırı miktardaki otomatik taleplerin engellendiği kontroller yer alır.

#	Açıklama	Seviye
2.4.1	Uygulama işlevlerine yönelik aşırı miktardaki çağrılara karşı; veri sızdırma, anlamsız veri üretme, kota tüketme, hız sınırının aşılması, hizmet reddi (DoS) ya da maliyetli kaynakların aşırı kullanımı gibi risklere karşı otomasyon karşıtı kontrollerin bulunduğu doğrulanmalıdır.	2
2.4.2	İş mantığı akışlarının, aşırı hızlı işlem gönderimlerini önleyecek şekilde gerçekçi insan zamanlaması gerektirdiği doğrulanmalıdır.	3

Referanslar

Daha fazla bilgi için:

- OWASP Web Security Testing Guide 4.2: Input Validation Testing
- OWASP Web Security Testing Guide 4.2: Business Logic Testing
- Anti-automation can be achieved in many ways, including the use of the OWASP Automated Threats to Web Applications
- OWASP Input Validation Cheat Sheet
- JSON Schema

V3 Web Ön Uç (Frontend) Güvenliği

Kontrol Amacı

Bu kategori, bir web frontend'i üzerinden gerçekleştirilen saldırılara karşı koruma sağlamak amacıyla belirlenmiş gereksinimlere odaklanır. Bu gereksinimler, makineden makineye çözümler için geçerli değildir.

V3.1 Web Frontend Güvenlik Dokümantasyonu

Bu bölüm, uygulamanın dokümantasyonunda belirtilmesi gereken tarayıcı güvenlik özelliklerini tanımlar.

#	Açıklama	Seviye
3.1.1	Uygulama dokümantasyonunda, uygulamayı kullanan tarayıcıların desteklemesi gereken güvenlik özelliklerinin (ör. HTTPS, HTTP Strict Transport Security (HSTS), Content Security Policy (CSP) ve diğer ilgili HTTP güvenlik mekanizmaları) tanımlandığı doğrulanmalıdır. Ayrıca bu özelliklerin mevcut olmadığı durumlarda uygulamanın nasıl davranması gerektiği (ör. kullanıcıyı uyarmak veya erişimi engellemek) de tanımlanmalıdır.	3

V3.2 İçeriğin Yanlış Yorumlanması

İçerik veya işlevin yanlış bir bağlamda sunulması, kötü amaçlı içeriğin çalıştırılmasına veya görüntülenmesine neden olabilir.

#	Açıklama	Seviye
3.2.1	Tarayıcıların HTTP yanıtlarında içerik veya işlevi yanlış bir bağlamda sunmalarını önlemek amacıyla güvenlik kontrollerinin uygulandığı doğrulanmalıdır (ör. API, kullanıcı tarafından yüklenen dosya veya başka bir kaynağın doğrudan çağrılması durumunda). Olası kontroller arasında; yalnızca HTTP istek başlıklarının (ör. Sec-Fetch-*) doğru bağlamı gösterdiği durumlarda içeriğin sunulması, Content-Security-Policy başlığında “sandbox” yönergesinin kullanılması veya Content-Disposition başlığında “attachment”biçiminin kullanılması yer alabilir.	1
3.2.2	HTML yerine metin olarak görüntülenmesi amaçlanan içeriğin, HTML veya JavaScript gibi içeriklerin istenmeden çalıştırılmasını önleyecek şekilde “createTextNode”veya “textContent”gibi güvenli işleme fonksiyonlarıyla işlendiği doğrulanmalıdır.	1

#	Açıklama	Seviye
3.2.3	Uygulamanın istemci tarafı JavaScript kullanırken açık değişken tanımlamaları yaparak, sıkı tür kontrolü uygulayarak, global değişkenleri document nesnesine kaydetmekten kaçınarak ve ad alanı izolasyonu sağlayarak DOM clobbering'den kaçındığı doğrulanmalıdır.	3

V3.3 Çerez Yapılandırması

Bu bölüm, hassas çerezlerin güvenli biçimde yapılandırılması için gereksinimleri tanımlar. Amaç, çerezlerin gerçekten uygulama tarafından oluşturulduğuna dair daha yüksek düzeyde güvence sağlamak ve içeriklerinin sızmasını veya uygunsuz şekilde değiştirilmesini önlemektir.

#	Açıklama	Seviye
3.3.1	Çerezlerin 'Secure'niteliğine sahip olduğu ve eğer çerez adı '__Host-'ön eki ile başlamıyorsa, '__Secure-'ön ekinin kullanıldığı doğrulanmalıdır.	1
3.3.2	Kullanıcı arayüzü kandırma saldırılarına ve tarayıcı tabanlı istek sahteciliği saldırılarına (CSRF) karşı koruma sağlamak amacıyla, her çerezin 'SameSite'niteliğinin kullanım amacına uygun şekilde ayarlandığı doğrulanmalıdır.	2
3.3.3	Çerez adı, diğer sunucularla paylaşılmak üzere açıkça tasarlanmadığı sürece '__Host-' ön eki ile tanımlanmalıdır.	2

#	Açıklama	Seviye
3.3.4	Bir çerez değeri istemci tarafı betikleri tarafından erişilememesi gereken bir veri içeriyorsa (ör. oturum token'ı), çerezin 'HttpOnly'niteliğine sahip olduğu ve aynı değerin (ör. oturum token'ı) yalnızca 'Set-Cookie'başlığı ile istemciye iletildiği doğrulanmalıdır.	2
3.3.5	Uygulama bir çerez oluştururken, çerez adı ve değerinin toplam uzunluğunun 4096 baytı aşmadığı doğrulanmalıdır. Çok büyük çerezler tarayıcı tarafından saklanmaz ve isteklerle gönderilmez; bu da çereze bağlı çalışan uygulama işlevlerinin bozulmasına yol açabilir.	3

V3.4 Tarayıcı Güvenlik Mekanizması Başlıkları (Header)

Bu bölüm, tarayıcıların uygulamadan gelen yanıtları işlerken güvenlik özelliklerini etkinleştirmesi ve kısıtlamaları uygulaması için HTTP yanıtlarında hangi güvenlik header'larının ayarlanması gerektiğini açıklar.

#	Açıklama	Seviye
3.4.1	Tüm yanıtların, HTTP Strict Transport Security (HSTS) politikasını zorunlu kılmak için Strict-Transport-Security header alanını içerdiği doğrulanmalıdır. En az 1 yıl süreli bir max-age tanımlanmalı ve Seviye 2 ve üzeri için bu politika tüm alt alan adlarını da kapsamalıdır.	1
3.4.2	Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin header'ı, uygulama tarafından sabit bir değer olarak tanımlanmalı veya gelen Origin header değeri güvenilir origin' lerden oluşan bir izinli listeye karşı doğrulanmalıdır. ' Access-Control-Allow-Origin: *' kullanılması gerekiyorsa, yanıtın hassas bilgi içermediği doğrulanmalıdır.	1

#	Açıklama	Seviye
3.4.3	HTTP yanıtlarında Content-Security-Policy header'ı bulunduğu ve bu header'ın yalnızca güvenilen içeriklerin veya kaynakların yüklenmesini ve çalıştırılmasını sağlayan yönergeler içerdiği doğrulanmalıdır. Minimum olarak "global policy", object-src 'none', base-uri 'none'yönergelerini içermeli ve izinli liste, nonce veya hash temelli bir yapı içermelidir. Seviye 3 uygulamalarında her yanıt için nonce veya hash içeren özel bir politika tanımlanmalıdır.	2
3.4.4	Tüm HTTP yanıtlarının 'X-Content-Type-Options: nosniff'header'ını içerdiği doğrulanmalıdır. Bu, tarayıcıya içerik türünü tahmin etmeyi bırakmasını ve yanıtın Content-Type header'ında belirtilen MIME türünün, beklenen kaynak türüyle eşleşmesini zorunlu kılar. Ayrıca bu, tarayıcıda CORB (Cross-Origin Read Blocking) özelliğini etkinleştirir.	2

#	Açıklama	Seviye
3.4.5	Uygulamanın, ‘Referer’başlığı yoluyla üçüncü taraflara teknik olarak hassas verilerin sızmasını önlemek için bir ‘referrer policy’belirlediği doğrulanmalıdır. Bu, Referrer-Policy HTTP header’ ıyla veya HTML öğe nitelikleriyle (element attributes) sağlanabilir. Sızabilecek hassas bilgiler, URL’deki yol ve sorgu parametreleri ile sınırlı uygulamalarda hostname bilgisini içerebilir.	2
3.4.6	Uygulamanın, Content-Security-Policy header’ı altında ‘frame-ancestors’yönergesini her HTTP yanıtında tanımladığı ve bu sayede içeriklerin varsayılan olarak başka sayfalara gömülmesini engellediği doğrulanmalıdır. Gerekli durumlarda özel olarak izin verildiğinden emin olunmalıdır. ‘X-Frame-Options’ header’ı, tarayıcılar tarafından desteklenmesine rağmen, artık güncel değildir ve güvenlik amacıyla kullanılmamalıdır.	2
3.4.7	Content-Security-Policy header’ında ihlal bildirimlerinin gönderileceği bir adresin tanımlandığı doğrulanmalıdır.	3

#	Açıklama	Seviye
3.4.8	Bir belgenin işlenmesini ve sunulmasını başlatan tüm HTTP yanıtlarında (örneğin 'text/html'tipi yanıtlar), 'Cross-Origin-Opener-Policy' header'ı 'same-origin'veya 'same-origin-allow-popups' yönergeleriyle birlikte yer almalıdır. Bu, tabnabbing ve frame counting gibi, pencere nesnelerine paylaşımlı erişimi sömüren saldırılara karşı koruma sağlar.	3

V3.5 Tarayıcı Menşei Ayırımı

Sunucu tarafında hassas işlevlere yönelik bir isteği kabul ederken, uygulamanın, isteğin uygulamanın kendisi veya güvenilir bir tarafça başlatıldığından ve bir saldırgan tarafından taklit edilmediğinden emin olması gerekir.

Bu bağlamda hassas işlevsellik, kimliği doğrulanmış ve doğrulanmamış kullanıcılar için form gönderilerinin kabul edilmesini (kimlik doğrulama isteği gibi), durum değiştirme işlemlerini veya kaynak gerektiren işlevselliği (veri dışa aktarma gibi) içerebilir.

Buradaki temel korumalar, JavaScript için Same Origin Policy ve çerezler için SameSite mantığı gibi tarayıcı güvenlik politikalarıdır. Bir diğer yaygın koruma ise CORS ön kontrol mekanizmasıdır. Bu mekanizma, farklı bir kaynaktan çağrılmak üzere tasarlanmış uç noktalar için kritik olacaktır, ancak farklı bir kaynaktan çağrılmak üzere tasarlanmamış uç noktalar için de yararlı bir istek sahteciliği önleme mekanizması olabilir.

#	Açıklama	Seviye
3.5.1	Uygulama, hassas işlevselliğe yönelik izinsiz cross-origin istekleri önlemek için CORS preflight mekanizmasına güvenmiyorsa, bu tür isteklerin gerçekten uygulamadan geldiği doğrulanmalıdır. Bu, anti-forgery token kullanımı veya CORS tarafından güvenli kabul edilmeyen özel HTTP header'larının zorunlu kılınmasıyla yapılabilir. Bu, tarayıcı tabanlı istek sahteciliği (CSRF) saldırılarına karşı bir savunmadır.	1
3.5.2	Uygulama, hassas işlevlerin izinsiz cross-origin çağrılmasını önlemek için CORS preflight mekanizmasına güveniyorsa, bu işlevlerin bir CORS-preflight isteğini tetiklemeden çağrılamadığı doğrulanmalıdır. Bu, 'Origin've 'Content-Type'header'larının doğrulanmasını veya CORS-safelisted olmayan bir header'ın ve alanın kullanılmasını gerektirebilir.	1

#	Açıklama	Seviye
3.5.3	Hassas işlevselliğe yönelik HTTP isteklerinin 'POST', 'PUT' , 'PATCH'veya 'DELETE'gibi uygun HTTP yöntemlerini kullandığı ve HTTP spesifikasyonuna göre "güvenli"olan 'GET', 'OPTIONS' veya 'HEAD'gibi yöntemlerle çağrılamadığı doğrulanmalıdır. Alternatif olarak 'Sec-Fetch-*' header'larının sıkı şekilde doğrulanması da uygunsuz cross-origin, navigasyon veya medya yükleme isteklerini ayırt etmek için kullanılabilir.	1
3.5.4	Farklı uygulamaların farklı hostname'lerde barındırıldığı ve böylece Same-Origin Policy tarafından sağlanan kısıtlamalardan (örneğin çerez erişim sınırları, kaynaklar arası etkileşim kontrolleri) faydalanıldığı doğrulanmalıdır.	2
3.5.5	postMessage arayüzü üzerinden alınan iletilerin, mesajın kökeni güvenilir değilse veya mesaj sözdizimi geçersizse reddedildiği doğrulanmalıdır.	2
3.5.6	Uygulamanın herhangi bir yerinde JSONP işlevselliğinin etkin olmadığı doğrulanmalıdır. Bu, XSSI (Cross-Site Script Inclusion) saldırılarını önlemek içindir.	3

#	Açıklama	Seviye
3.5.7	Yetkilendirme gerektiren verilerin, JavaScript dosyaları gibi betik kaynaklarının yanıtlarında yer almadığı, XSSI saldırılarını önlemek için doğrulanmalıdır.	3
3.5.8	Kimliği doğrulanmış kaynakların (örneğin görseller, videolar, betikler ve belgeler) yalnızca gerçekten amaçlandığında kullanıcı adına yüklenebildiği veya gömülebildiği doğrulanmalıdır. Bu, isteğin uygun olmayan bir çapraz kaynak çağrısından kaynaklanmadığından emin olmak için Sec-Fetch-* HTTP istek başlığı alanlarının sıkı bir şekilde doğrulanmasıyla veya tarayıcıya döndürülen içeriği engelleme talimatı vermek için kısıtlayıcı bir Cross-Origin-Resource-Policy HTTP yanıt başlığı alanı ayarlayarak gerçekleştirilebilir.	3

V3.6 Harici Kaynak Bütünlüğü

Bu bölüm, içeriklerin üçüncü taraf sitelerde güvenli şekilde barındırılmasıyla ilgili rehberlik sunar.

#	Açıklama	Seviye
3.6.1	JavaScript kütüphaneleri, CSS veya web fontları gibi istemci tarafı varlıkların yalnızca statik ve sürümlü olması, varlığın bütünlüğünü doğrulamak için Subresource Integrity (SRI) kullanılması durumunda harici olarak (örneğin, bir İçerik Dağıtım Ağı üzerinde) barındırıldığı doğrulanmalıdır. Bu mümkün değilse, her bir kaynak için bunu gerekçelendiren belgelenmiş bir güvenlik kararı bulunmalıdır.	3

V3.7 Diğer Tarayıcı Güvenliği Hususları

Bu bölüm, istemci tarafı tarayıcı güvenliği için gerekli olan çeşitli diğer güvenlik kontrolleri ve modern tarayıcı güvenlik özelliklerini içerir.

#	Açıklama	Seviye
3.7.1	Uygulamanın yalnızca hâlâ desteklenen ve güvenli kabul edilen istemci tarafı teknolojileri kullandığı doğrulanmalıdır. Bu gereksinimi karşılamayan teknolojilere örnek olarak NSAPI eklentileri, Flash, Shockwave, ActiveX, Silverlight, NACL veya istemci tarafı Java applet'leri verilebilir.	2
3.7.2	Uygulamanın, yalnızca hedef adres bir izinli listede yer alıyorsa kullanıcıyı otomatik olarak farklı bir ana makine adına veya uygulamanın kontrolünde olmayan bir etki alanına yönlendirdiği doğrulanmalıdır.	2
3.7.3	Uygulamanın, kullanıcı uygulamanın kontrolü dışındaki bir URL'ye yönlendirildiğinde, yönlendirme hakkında bir bildirim gösterdiği ve kullanıcının bu gezintiyi iptal edebilmesi için bir seçenek sunduğu doğrulanmalıdır.	3

#	Açıklama	Seviye
3.7.4	Uygulamanın en üst düzey etki alanının (örneğin, site.tld) HTTP Strict Transport Security (HSTS) için genel preload listesine eklendiği doğrulanmalıdır. Bu, uygulamanın TLS kullanımının ana tarayıcılar tarafından doğrudan tanınmasını ve yalnızca Strict-Transport-Security yanıt header'ına bağlı kalınmamasını sağlar.	3
3.7.5	Uygulamaya erişen tarayıcının beklenen güvenlik özelliklerini desteklememesi durumunda, uygulamanın dokümantasyona uygun şekilde davrandığı (örneğin kullanıcıyı uyarma veya erişimi engelleme) doğrulanmalıdır.	3

Referanslar

Daha fazla bilgi için:

- Set-Cookie __Host- prefix details
- OWASP Content Security Policy Cheat Sheet
- Exploiting CORS misconfiguration for BitCoins and Bounties
- Sandboxing third-party components
- OWASP Secure Headers Project
- OWASP Cross-Site Request Forgery Prevention Cheat Sheet
- HSTS Browser Preload List submission form

V4 API ve Web Servisi

Kontrol Amacı

Web tarayıcıları veya diğer istemciler tarafından kullanılmak üzere API'ler sunan uygulamalara özgü bazı ek değerlendirmeler bulunmaktadır (genellikle JSON, XML veya GraphQL kullanılır). Bu bölüm, uygulanması gereken ilgili güvenlik yapılandırmaları ve mekanizmaları kapsar.

Kimlik doğrulama, oturum yönetimi ve girdi doğrulama gibi diğer bölümlerde ele alınan konuların API'ler için de geçerli olduğunu unutmayın; bu nedenle bu bölüm bağlamından bağımsız düşünülemez veya tek başına test edilemez.

V4.1 Genel Web Servisi Güvenliği

Bu bölüm, genel web servisi güvenliğiyle ilgili değerlendirmeleri ve temel web servisi hijyen uygulamalarını ele alır.

#	Açıklama	Seviye
4.1.1	İleti gövdesi içeren her HTTP yanıtında, yanıtın gerçek içeriğiyle eşleşen bir Content-Type header alanı bulunduğu ve güvenli karakter kodlamasını (örneğin, UTF-8, ISO-8859-1) belirtmek üzere charset parametresi içerdiği doğrulanmalıdır. Bu işlem, IANA Medya Türleri'nden "text/", "/+xml"ve "/xml"gibi içerikler için geçerlidir.	1
4.1.2	Yalnızca kullanıcı arayüzüne yönelik uç noktaların (manuel olarak web tarayıcısı ile erişilmesi amaçlanan) otomatik olarak HTTP'den HTTPS'e yönlendirme yaptığı ve diğer hizmetlerin veya uç noktaların şeffaf yönlendirme gerçekleştirmediği doğrulanmalıdır. Bu, istemcinin farkında olmadan şifrelenmemiş HTTP istekleri göndermesi ve bu isteklerin otomatik olarak HTTPS'e yönlendirilmesi sonucunda hassas verilerin sızmasının fark edilmemesi riskini önlemek içindir.	2
4.1.3	Uygulama tarafından kullanılan ve yük dengeleyici, web proxy veya frontend için backend hizmeti gibi bir ara katman tarafından ayarlanan herhangi bir HTTP başlık alanının, son kullanıcı tarafından geçersiz kılınamayacağı doğrulanmalıdır. Örnek başlıklar arasında X-Real-IP, X-Forwarded-* veya X-User-ID yer alabilir.	2
4.1.4	Uygulama veya API tarafından açıkça desteklenen HTTP yöntemlerinin (preflight istekleri sırasında OPTIONS dahil) dışında hiçbir yöntemin kullanılmadığı ve kullanılmayan yöntemlerin engellendiği doğrulanmalıdır.	3
4.1.5	Çok hassas istekler veya birçok sistemden geçen işlemler için, taşıma katmanı güvenliğinin üzerine ek bir güvence sağlamak amacıyla, her iletiye özel dijital imzaların kullanıldığı doğrulanmalıdır.	3

V4.2 HTTP İleti Yapısı Doğrulama

Bu bölüm, HTTP ileti yapısının ve başlık alanlarının doğrulanma biçimini açıklar. Amaç, istek kaçakçılığı, yanıt bölme, başlık enjeksiyonu ve aşırı uzun HTTP iletileri aracılığıyla hizmet reddi gibi saldırıları önlemektir.

Bu gereksinimler, genel HTTP ileti işleme ve üretimi için geçerlidir; ancak farklı HTTP sürümleri arasında HTTP iletileri dönüştürülürken özellikle önemlidir.

#	Açıklama	Seviye
4.2.1	Tüm uygulama bileşenlerinin (yük dengeleyiciler, güvenlik duvarları ve uygulama sunucuları dahil) HTTP sürümüne uygun mekanizmaları kullanarak gelen HTTP iletilerinin sınırlarını belirlediği ve bu sayede HTTP istek kaçakçılığına karşı koruma sağlandığı doğrulanmalıdır. HTTP/1.x sürümünde Transfer-Encoding başlık alanı varsa, RFC 2616'ya göre Content-Length başlığı göz ardı edilmelidir. HTTP/2 veya HTTP/3 kullanıldığında, Content-Length başlığı varsa, alıcının bu değerin DATA çerçevelerinin uzunluğuyla tutarlı olduğunu doğrulaması gerekir.	2
4.2.2	HTTP iletileri üretilirken, Content-Length başlığının, HTTP protokolünün çerçeveleme yöntemine göre belirlenen içerik uzunluğu ile çelişmediği doğrulanmalıdır. Bu, istek kaçakçılığı saldırılarını önlemek için gereklidir.	3
4.2.3	Uygulamanın, yanıt bölme ve header enjeksiyonu saldırılarını önlemek amacıyla, bağlantıya özel başlık alanları (örneğin Transfer-Encoding) içeren HTTP/2 veya HTTP/3 iletilerini göndermediği ve kabul etmediği doğrulanmalıdır.	3
4.2.4	Uygulamanın yalnızca başlık alanları ve değerleri CR (`\r`), LF (`\n`) veya CRLF (`\r\n`) karakter dizileri içermeyen HTTP/2 ve HTTP/3 isteklerini kabul ettiği doğrulanmalıdır. Bu, başlık enjeksiyonu saldırılarını önlemek içindir.	3
4.2.5	Uygulama (backend veya frontend) istek oluşturuyor ve gönderiyorsa, URI'ler (örneğin API çağrıları için) veya HTTP istek başlıkları (örneğin Authorization veya Cookie) gibi bileşenlerin, alıcı sistemler tarafından kabul edilemeyecek kadar uzun olmamasını sağlamak amacıyla doğrulama, temizleme veya diğer mekanizmaları kullandığı doğrulanmalıdır. Bu tür çok uzun istekler (örneğin çok uzun bir Cookie başlığı) gönderildiğinde sunucunun sürekli hata durumu döndürmesiyle hizmet reddi oluşabilir.	3

V4.3 GraphQL

GraphQL, veri açısından zengin istemcilerin çeşitli backend servislerine sıkı sıkıya bağlı olmadan oluşturulması için giderek daha yaygın hale gelen bir yöntemdir. Bu bölüm, GraphQL ile ilgili güvenlik değerlendirmelerini kapsar.

#	Açıklama	Seviye
4.3.1	Maliyetli ve iç içe geçmiş sorgular sonucu oluşabilecek GraphQL veya veri katmanı ifadelerine yönelik Hizmet Reddi (DoS) saldırılarını önlemek için, sorgu izin listesi (allowlist), derinlik sınırlandırması, sorgu sayısı sınırlandırması veya sorgu maliyeti analizi kullanıldığı doğrulanmalıdır.	2
4.3.2	GraphQL introspection (iç yapı sorgulama) sorgularının, GraphQL API'nin üçüncü taraflar tarafından kullanılmasına yönelik olmadığı sürece, üretim ortamında devre dışı bırakıldığı doğrulanmalıdır.	2

V4.4 WebSocket

WebSocket, tek bir TCP bağlantısı üzerinden eşzamanlı çift yönlü iletişim sağlayan bir iletişim protokolüdür. 2011 yılında IETF tarafından RFC 6455 olarak standartlaştırılmıştır ve HTTP'den farklıdır, ancak 443 ve 80 numaralı HTTP portları üzerinden çalışacak şekilde tasarlanmıştır.

Bu bölüm, özellikle bu gerçek zamanlı iletişim kanalını hedef alan iletişim güvenliği ve oturum yönetimi ile ilgili saldırılara karşı korunmak için gerekli olan güvenlik gereksinimlerini sunar.

#	Açıklama	Seviye
4.4.1	Tüm WebSocket bağlantılarında TLS üzerinden WebSocket (WSS) kullanıldığı doğrulanmalıdır.	1
4.4.2	WebSocket'in başlangıçtaki HTTP el sıkışması (handshake) sırasında, Origin başlık alanının uygulama için izin verilen origin listesiyle karşılaştırılarak kontrol edildiği doğrulanmalıdır.	2
4.4.3	Uygulamanın standart oturum yönetimi kullanılamıyorsa, bunun yerine ilgili Oturum Yönetimi güvenlik gereksinimlerine uygun özel token'ların kullanıldığı doğrulanmalıdır.	2
4.4.4	HTTPS oturumu WebSocket kanalına geçerken, özel WebSocket oturum yönetim token'larının önceki şekilde kimliği doğrulanmış HTTPS oturumu aracılığıyla elde edildiği veya doğrulandığı doğrulanmalıdır.	2

Referanslar

Daha fazla bilgi için:

- OWASP REST Security Cheat Sheet
- Resources on GraphQL Authorization from graphql.org and Apollo.
- WSTG - v4.2 | GraphQL Testing

- WSTG - v4.1 | OWASP Foundation

V5 Dosya İşleme

Kontrol Amacı

Dosya kullanımı, hizmet reddi, yetkisiz erişim ve depolama alanının tükenmesi gibi uygulama için çeşitli riskler oluşturabilir. Bu bölüm, bu riskleri ele almak için gereken gereksinimleri içerir.

V5.1 Dosya İşleme Dokümantasyonu

Bu bölüm, uygulama tarafından kabul edilen dosyaların beklenen özelliklerinin belgelenmesini gerektirir. Bu, ilgili güvenlik kontrollerinin geliştirilmesi ve doğrulanması için gerekli bir ön koşuldur.

#	Açıklama	Seviye
5.1.1	Dokümantasyonun, her yükleme özelliği için izin verilen dosya türlerini, beklenen dosya uzantılarını ve maksimum boyutu (açılmış hali dahil) tanımladığı doğrulanmalıdır. Ayrıca, dosyaların son kullanıcılar için nasıl güvenli hâle getirileceği (örneğin kötü amaçlı bir dosya tespit edildiğinde uygulamanın nasıl davranacağı) gibi bilgilerin de belgede yer aldığı doğrulanmalıdır.	2

V5.2 Dosya Yükleme ve İçerik

Dosya yükleme işlevi, güvenilmeyen dosyaların birincil kaynağıdır. Bu bölüm, bu dosyaların varlığının, hacminin veya içeriğinin uygulamaya zarar vermemesini sağlamak için gereken gereksinimleri açıklar.

#	Açıklama	Seviye
5.2.1	Uygulamanın, yalnızca performans kaybına veya hizmet reddine yol açmayacak boyuttaki dosyaları kabul ettiği doğrulanmalıdır.	1

#	Açıklama	Seviye
5.2.2	Uygulama bir dosyayı, doğrudan veya bir arşiv (ör. zip dosyası) içinde kabul ettiğinde, dosya uzantısının beklenen uzantıyla eşleşip eşleşmediğini kontrol ettiği ve içeriğin uzantı tarafından temsil edilen türle uyumlu olduğunu doğruladığı kontrol edilmelidir. Bu; ilk 'magic byte'kontrolü, görüntü yeniden yazımı ve dosya içeriği doğrulama için özel kütüphanelerin kullanımı gibi yöntemleri içerir. Seviye 1 için bu, yalnızca iş veya güvenlik kararlarında kullanılan dosyalarla sınırlı olabilir. Seviye 2 ve üzeri için tüm dosyalar bu kontrole tabi olmalıdır.	1
5.2.3	Sıkıştırılmış dosyaların (ör. zip, gz, docx, odt) açılmadan önce, izin verilen maksimum açılmış boyuta ve maksimum dosya sayısına göre kontrol edildiği doğrulanmalıdır.	2
5.2.4	Her kullanıcı için dosya boyutu kotası ve maksimum dosya sayısının uygulandığı doğrulanmalıdır. Böylece tek bir kullanıcının çok sayıda veya aşırı büyük dosya yükleyerek depolama alanını doldurması engellenir.	3
5.2.5	Uygulamanın, içinde sembolik bağlantılar (symlink) bulunan sıkıştırılmış dosyaların yüklenmesine yalnızca bu durum özel olarak gerekli olduğunda izin verdiği doğrulanmalıdır. Bu durumda, sembolik bağlantı verilebilecek dosyaların bir izinli listesi uygulanmalıdır.	3
5.2.6	Uygulamanın, piksel boyutu izin verilen maksimum değeri aşan görüntü dosyalarının yüklenmesini reddettiği doğrulanmalıdır. Bu, pixel flood saldırılarını önlemek için gereklidir.	3

V5.3 Dosya Depolama

Bu bölüm, yüklenen dosyaların uygunsuz şekilde çalıştırılmasını önlemeye, tehlikeli içeriğin tespitine ve güvenilmeyen verilerle dosya depolama konumlarının kontrol edilmesinin engellenmesine yönelik gereksinimleri içerir.

#	Açıklama	Seviye
5.3.1	Güvenilmeyen girdilerle yüklenen veya oluşturulan ve herkese açık klasörlerde depolanan dosyaların, bir HTTP isteğiyle doğrudan erişildiğinde sunucu taraflı program kodu olarak çalıştırılmadığı doğrulanmalıdır.	1

#	Açıklama	Seviye
5.3.2	Uygulamanın, dosya işlemleri için dosya yollarını oluştururken kullanıcı tarafından gönderilen dosya adlarını değil, dahili olarak oluşturulan veya güvenilir verileri kullandığı; eğer kullanıcı tarafından gönderilen dosya adları veya meta veriler kullanılacaksa, sıkı doğrulama ve temizleme işlemlerinin uygulandığı doğrulanmalıdır. Bu, yol geçişi (path traversal), yerel veya uzak dosya dahil etme (LFI, RFI) ve sunucu tarafı istek sahteciliği (SSRF) saldırılarına karşı koruma sağlar.	1
5.3.3	Sunucu tarafında gerçekleşen dosya işlemlerinin (örneğin dosya açma/decompress işlemi), kullanıcı tarafından sağlanan yol bilgilerini yok saydığı ve zip slip gibi güvenlik açıklarını önlediği doğrulanmalıdır.	3

V5.4 Dosya İndirme

Bu bölüm, indirilebilir dosyaların sunulmasında ortaya çıkabilecek riskleri azaltmaya yönelik gereksinimleri içerir. Bunlar; yol geçişi (path traversal), enjeksiyon saldırıları ve dosyanın tehlikeli içerik barındırıp barındırmadığına ilişkin kontrolleri kapsar.

#	Açıklama	Seviye
5.4.1	Uygulamanın, kullanıcı tarafından gönderilen dosya adlarını (ör. JSON, JSONP veya URL parametresinde) doğruladığı veya göz ardı ettiği ve yanıtın Content-Disposition başlık alanında bir dosya adı belirttiği doğrulanmalıdır.	2
5.4.2	Sunulan dosya adlarının (örneğin HTTP yanıt header'larında veya e-posta eklerinde), belge yapısını korumak ve enjeksiyon saldırılarını önlemek için kodlandığı veya temizlendiği (ör. RFC 6266'ya uygun) doğrulanmalıdır.	2
5.4.3	Güvenilmeyen kaynaklardan elde edilen dosyaların, bilinen kötü amaçlı içeriklerin sunulmasını önlemek amacıyla antivirüs tarayıcılarıyla tarandığı doğrulanmalıdır.	2

Referanslar

Daha fazla bilgi için:

- File Extension Handling for Sensitive Information
- Example of using symlinks for arbitrary file read
- Explanation of "Magic Bytes"from Wikipedia

V6 Kimlik Doğrulama

Kontrol Amacı

Kimlik doğrulama, bir kişinin ya da cihazın kimliğini kanıtlama veya doğrulama sürecidir. Bu süreç; bir kişi ya da cihaz hakkında yapılan iddiaların doğrulanmasını, taklit girişimlerine karşı dayanıklılığı ve parolaların ele geçirilmesine ya da dinlenmesine karşı korunmayı kapsar.

NIST SP 800-63, özellikle ABD kurumları ve bu kurumlarla etkileşimde bulunanlar için geçerli olmakla birlikte, dünya çapında kuruluşlar için değerli olan modern ve kanıta dayalı bir standarttır.

Bu bölümdeki birçok gereksinim, söz konusu standardın ikinci bölümü olan NIST SP 800-63B "Dijital Kimlik Rehberi –Kimlik Doğrulama ve Yaşam Döngüsü Yönetimi"bölümünden esinlenmiştir. Ancak bu bölüm, yaygın tehditlere ve sıkça suistimal edilen kimlik doğrulama zayıflıklarına odaklanır; standardın tüm yönlerini kapsamaya çalışmaz. Tam NIST SP 800-63 uyumu gerekiyorsa, doğrudan bu belgeye başvurulmalıdır.

Ayrıca, NIST SP 800-63 terimleriyle bu bölümde kullanılan terimler zaman zaman farklılık gösterebilir. Bu bölümde, anlaşılırlığı artırmak amacıyla daha yaygın kullanılan terimler tercih edilmiştir.

Daha gelişmiş uygulamaların ortak bir özelliği, çeşitli risk faktörlerine bağlı olarak kimlik doğrulama aşamalarının uyarlanabilir şekilde yapılandırılabilmesidir. Bu özellik, yetkilendirme kararlarıyla da ilgili olduğundan "Yetkilendirme"bölümünde ele alınmıştır.

V6.1 Kimlik Doğrulama Dokümantasyonu

Bu bölüm, bir uygulamada kimlik doğrulama ile ilgili tutulması gereken dokümantasyon gereksinimlerini içerir. Bu, ilgili kimlik doğrulama kontrollerinin nasıl yapılandırılması gerektiğini uygulamak ve değerlendirmek açısından kritik öneme sahiptir.

#	Açıklama	Seviye
6.1.1	Kimlik bilgisi doldurma (credential stuffing) ve parola kaba kuvvet saldırılarına karşı oran sınırlama (rate limiting), otomasyon karşıtı önlemler ve uyarlanabilir yanıt gibi kontrollerin nasıl kullanıldığını açıklayan bir uygulama dokümantasyonu bulunduğu doğrulanmalıdır. Dokümantasyon, bu kontrollerin nasıl yapılandırıldığını ve kötü niyetli hesap kilitlemenin nasıl önlendiğini açıkça ortaya koymalıdır.	1
6.1.2	Parolalarda kullanılmasını önlemek amacıyla, bağlama özgü sözcüklerin (örneğin kurum adlarının permütasyonları, ürün adları, sistem tanımlayıcıları, proje kod adları, departman veya rol adları vb.) bulunduğu belgelenmiş bir liste mevcut olduğu doğrulanmalıdır.	2

#	Açıklama	Seviye
6.1.3	Uygulama birden fazla kimlik doğrulama yolu içeriyorsa, bunların tümünün belgelenmiş olduğu ve her yol için tutarlı şekilde uygulanması gereken güvenlik kontrolleri ile kimlik doğrulama gücünün tanımlandığı doğrulanmalıdır.	2

V6.2 Parola Güvenliği

Parolalar, NIST SP 800-63 tarafından "Ezberlenmiş Sırlar"(Memorized Secrets) olarak tanımlanır ve parolalar, parola öbekleri, PIN'ler, desen kilitleri ya da "doğru kedinin veya başka bir görsel öğenin seçilmesi"gibi yöntemleri kapsar. Genellikle "bildiğiniz bir şey"olarak kabul edilirler ve tek faktörlü kimlik doğrulamada yaygın olarak kullanılırlar.

Bu nedenle, bu bölüm parolaların güvenli şekilde oluşturulmasını ve işlenmesini sağlamak amacıyla oluşturulmuş gereksinimleri içerir. Gereksinimlerin çoğu Seviye 1'dedir çünkü temel düzeyde en önemli olanlardır. Seviye 2 ve sonrasında, çok faktörlü kimlik doğrulama mekanizmaları gereklidir ve parola bu faktörlerden yalnızca biri olabilir.

Bu bölümdeki gereksinimler büyük ölçüde NIST's Guidance'ın § 5.1.1.2 kısmı ile ilişkilidir.

#	Açıklama	Seviye
6.2.1	Kullanıcı tarafından belirlenen parolaların en az 8 karakter uzunluğunda olması sağlanmalı, ancak 15 karakter ve üzeri uzunlukta parolalar olmaları şiddetle tavsiye edilir.	1
6.2.2	Kullanıcıların parolalarını değiştirebilmeleri sağlanmalıdır.	1
6.2.3	Parola değiştirme işlevinin hem mevcut hem de yeni parolayı gerektirdiği doğrulanmalıdır.	1
6.2.4	Kayıt veya parola değişimi sırasında girilen parolaların, uygulamanın parola politikasına (örneğin minimum uzunluk) uyan ilk 3000 yaygın parolaya karşı kontrol edildiği doğrulanmalıdır.	1
6.2.5	Her tür karakterin kullanılmasına izin verildiği ve büyük/küçük harf, sayı veya özel karakter şartlarının zorunlu tutulmadığı doğrulanmalıdır.	1
6.2.6	Parola giriş alanlarının, girdiyi maskelemek için "type=password"kullandığı doğrulanmalıdır. Uygulamalar, kullanıcının tüm maskelenmiş parolayı ya da son girilen karakteri geçici olarak görüntülemesine izin verebilir.	1
6.2.7	"Yapıştır"özelliğinin, tarayıcı parola yardımcılarının ve harici parola yöneticilerinin kullanımına izin verildiği doğrulanmalıdır.	1

#	Açıklama	Seviye
6.2.8	Uygulamanın, kullanıcıdan aldığı parolayı herhangi bir şekilde değiştirmeden, örneğin kesmeden veya harf büyüklüğünü değiştirmeden, tam olarak doğruladığı teyit edilmelidir.	1
6.2.9	En az 64 karakter uzunluğundaki parolalara izin verildiği doğrulanmalıdır.	2
6.2.10	Kullanıcının parolasının, yalnızca ele geçirildiği tespit edildiğinde veya kullanıcı tarafından yenilendiğinde geçerliliğini yitirdiği doğrulanmalıdır. Uygulama, belirli aralıklarla parola yenilenmesini zorunlu kılmamalıdır.	2
6.2.11	Tahmin edilmesi kolay parolaların oluşturulmasını önlemek amacıyla, bağlama özgü kelimelerin yer aldığı belgelenmiş listenin parola doğrulamada kullanıldığı teyit edilmelidir.	2
6.2.12	Kayıt veya parola değişikliği sırasında girilen parolaların, ele geçirilmiş parolalar listesine karşı kontrol edildiği doğrulanmalıdır.	2

V6.3 Genel Kimlik Doğrulama Güvenliği

Bu bölüm, kimlik doğrulama mekanizmalarının güvenliğine ilişkin genel gereksinimleri içerir ve farklı güvenlik seviyeleri için beklentileri ortaya koyar. Seviye 2 uygulamalarında çok faktörlü kimlik doğrulama (MFA) zorunlu olmalıdır. Seviye 3 uygulamalarında, donanım tabanlı, kanıtlanmış ve güvenli yürütme ortamında (Trusted Execution Environment - TEE) gerçekleştirilen kimlik doğrulama kullanılmalıdır. Bu, cihaza bağlı passkey'ler, eIDAS Yüksek Güvence Seviyesi (LoA High) zorunlu kimlik doğrulayıcılar, NIST Kimlik Doğrulayıcı Güvence Seviyesi 3 (AAL3) güvencesine sahip kimlik doğrulayıcılar veya eşdeğer mekanizmaları içerebilir.

Bu yaklaşım MFA konusunda oldukça katı bir duruş sergilese de, kullanıcıları korumak adına güvenlik seviyesini yükseltmek kritiktir. Bu gereksinimlerin gevşetilmesi yönündeki her girişim, kimlik doğrulama ile ilgili risklerin nasıl azaltılacağını açıkça ortaya koyan bir planla birlikte sunulmalıdır. Bu plan, NIST'in konuyla ilgili rehberleri ve araştırmalarını dikkate almalıdır.

Yayın tarihi itibarıyla, NIST SP 800-63 e-posta kullanımını bir kimlik doğrulama mekanizması olarak kabul edilemez olarak değerlendirmektedir (arşivlenmiş kopya).

Bu bölümdeki gereksinimler, NIST's Guidance'ın § 4.2.1, § 4.3.1, § 5.2.2 ve § 6.1.2 bölümleriyle ilgilidir.

#	Açıklama	Seviye
6.3.1	Kimlik bilgisi doldurma (credential stuffing) ve parola kaba kuvvet saldırıları gibi saldırıları önlemeye yönelik kontrollerin, uygulamanın güvenlik dokümantasyonuna göre uygulandığı doğrulanmalıdır.	1

#	Açıklama	Seviye
6.3.2	“root”, “admin”veya “sa”gibi varsayılan kullanıcı hesaplarının uygulamada yer almadığı veya devre dışı bırakıldığı doğrulanmalıdır.	1
6.3.3	Uygulamaya erişim için ya çok faktörlü bir kimlik doğrulama mekanizması ya da tek faktörlü mekanizmaların kombinasyonu kullanılmalıdır. Seviye 3 için, bu faktörlerden biri, oltalama saldırılarına karşı direnç sağlayan ve kimlik doğrulama niyetini bir kullanıcı etkileşimi (örneğin FIDO donanım anahtarında bir düğmeye basma veya mobil telefonda bir onay) gerektirerek doğrulayan donanım tabanlı bir kimlik doğrulayıcı olmalıdır. Bu gereksinimdeki herhangi bir esneklik için, gerekçesi belgelenmiş ve riskleri azaltıcı kapsamlı kontroller içeren bir plan gereklidir.	2
6.3.4	Uygulama birden fazla kimlik doğrulama yolu içeriyorsa, belgelendirilmemiş bir yol bulunmadığı ve tüm yollar için güvenlik kontrolleri ile kimlik doğrulama gücünün tutarlı şekilde uygulandığı doğrulanmalıdır.	2
6.3.5	Başarılı veya başarısız şüpheli kimlik doğrulama girişimlerinin kullanıcıya bildirildiği doğrulanmalıdır. Bu, olağandışı bir konum veya istemciden gelen girişimler, yalnızca bir faktörle yapılan kısmen başarılı doğrulamalar, uzun süreli inaktiflik sonrası yapılan girişimler veya birçok başarısız denemeden sonraki başarılı girişimler gibi durumları içerebilir.	3
6.3.6	E-postanın, tek faktörlü ya da çok faktörlü kimlik doğrulama mekanizması olarak kullanılmadığı doğrulanmalıdır.	3
6.3.7	Kimlik doğrulama detaylarında yapılan güncellemeler sonrasında (örneğin kimlik bilgisi sıfırlama, kullanıcı adı veya e-posta adresi değişiklikleri), kullanıcıların bilgilendirildiği doğrulanmalıdır.	3
6.3.8	Başarısız kimlik doğrulama girişimlerinden, geçerli kullanıcıların hata mesajları, HTTP yanıt kodları veya farklı yanıt süreleri gibi sonuçlara dayanarak tespit edilemediği doğrulanmalıdır. Bu koruma, kayıt ve “parolamı unuttum”işlevselliğinde de geçerli olmalıdır.	3

V6.4 Kimlik Doğrulama Faktörünün Yaşam Döngüsü ve Kurtarma

Kimlik doğrulama faktörleri; parolalar, yazılım tabanlı token’lar, donanım token’ları ve biyometrik cihazlar gibi öğeleri kapsar. Bu mekanizmaların yaşam döngüsünün güvenli şekilde yönetilmesi, bir uygulamanın güvenliği açısından kritik öneme sahiptir. Bu bölüm, bu mekanizmalarla ilgili gereksinimleri içerir.

Bu bölümdeki gereksinimler, büyük ölçüde NIST’s Guidance’ın § 5.1.1.2 veya § 6.1.2.3 bölümleri ile ilişkilidir.

#	Açıklama	Seviye
6.4.1	Sistem tarafından oluşturulan ilk parolaların veya etkinleştirme kodlarının güvenli bir şekilde rastgele üretildiği, mevcut parola politikasına uygun olduğu, kısa bir süre içinde veya ilk kullanım sonrasında geçerliliğini yitirdiği doğrulanmalıdır. Bu başlangıç sırları uzun vadeli parola olarak kullanılmasına izin verilmemelidir.	1
6.4.2	Parola ipuçlarının veya "gizli sorular"olarak bilinen, bilgiye dayalı kimlik doğrulama mekanizmalarının mevcut olmadığı doğrulanmalıdır.	1
6.4.3	Unutulmuş parolanın sıfırlanması için, etkin olan çok faktörlü kimlik doğrulama mekanizmalarını atlamayan güvenli bir sürecin uygulandığı doğrulanmalıdır.	2
6.4.4	Çok faktörlü kimlik doğrulama faktörlerinden biri kaybolursa, kayıt sırasında uygulanan kimlik kanıtlama seviyesiyle aynı seviyede kimlik doğrulama yapıldığı doğrulanmalıdır.	2
6.4.5	Süresi dolacak kimlik doğrulama mekanizmalarının yenilenmesine ilişkin talimatların, süresi dolmadan önce tamamlanacak şekilde yeterli sürede gönderildiği ve gerekiyorsa otomatik hatırlatıcıların yapılandırıldığı doğrulanmalıdır.	3
6.4.6	Yönetici yetkisi olan kullanıcıların, kullanıcının parola sıfırlama sürecini başlatabildiği ancak kullanıcının yeni parolasını belirleyemediği veya değiştiremediği doğrulanmalıdır. Bu, yönetici yetkisi olan kullanıcıların, kullanıcı parolasını bilmesini engeller.	3

V6.5 Çok Faktörlü Kimlik Doğrulama İçin Genel Gereksinimler

Bu bölüm, çeşitli çok faktörlü kimlik doğrulama yöntemleriyle ilgili genel yönergeleri sunar.

Bu mekanizmalar şunları içerir:

- Arama Sırları (Lookup Secrets)
- Zamana Dayalı Tek Kullanımlık Parolalar (TOTP)
- Bant Dışı (Out-of-Band) mekanizmalar

"Arama sırları", daha önceden oluşturulmuş gizli kod listeleridir. Bunlar, İşlem Yetkilendirme Numaraları (TAN), sosyal medya kurtarma kodları veya rastgele değerler içeren bir tablo şeklinde olabilir. Bu tür kimlik doğrulama mekanizmaları, "sahip olduğunuz bir şey"olarak değerlendirilir çünkü kodlar kasıtlı olarak ezberlenemez olacak şekilde tasarlanır ve bir yerde saklanmaları gerekir.

Zamana Dayalı Tek Kullanımlık Parolalar (TOTP), fiziksel veya yazılımsal token'lardır ve sürekli

değişen, sözde rastgele (pseudo-random) bir tek kullanımlık parola gösterir. Bu mekanizmalar da "sahip olduğunuz bir şey"kategorisindedir. Çok faktörlü TOTP'ler tek faktörlü TOTP'lere benzer, ancak nihai Tek Kullanımlık Parolayı (OTP) oluşturmak için geçerli bir PIN kodu, biyometrik kilit açma, USB takma, NFC eşleştirme veya bazı ek değerlerin (işlem imzalama hesaplayıcıları gibi) girilmesini gerektirir.

Bant dışı mekanizmalar bir sonraki bölümde açıklanacaktır.

Bu bölümlerdeki gereklilikler çoğunlukla NIST's Guidance'ın § 5.1.2, § 5.1.3, § 5.1.4.2, § 5.1.5.2, § 5.2.1 ve § 5.2.3 kısımları ile ilgilidir.

#	Açıklama	Seviye
6.5.1	Arama sırları, bant dışı kimlik doğrulama istekleri veya kodları ve zamana dayalı tek kullanımlık parolaların (TOTP) yalnızca bir kez kullanılabildiği doğrulanmalıdır.	2
6.5.2	Uygulamanın backend tarafında saklanırken, 112 bitten az entropiye sahip (örneğin 19 rastgele harf-rakam karakteri veya 34 rastgele rakam) arama sırlarının, 32-bit rastgele bir salt içeren onaylı bir parola saklama algoritmasıyla hash'lendiği doğrulanmalıdır. Gizli bilginin entropisi 112 bit veya daha fazlaysa standart bir hash fonksiyonu kullanılabilir.	2
6.5.3	Arama sırları, bant dışı kimlik doğrulama kodları ve zamana dayalı tek kullanımlık parola seed'lerinin, öngörülebilir değerlerden kaçınmak için Kriptografik Olarak Güvenli Sözde Rastgele Sayı Üreteci (CSPRNG) ile oluşturulduğu doğrulanmalıdır.	2
6.5.4	Arama sırlarının ve bant dışı kimlik doğrulama kodlarının en az 20 bit entropiye (genellikle 4 rastgele harf-rakam karakteri veya 6 rastgele rakam) sahip olduğu doğrulanmalıdır.	2
6.5.5	Bant dışı kimlik doğrulama isteklerinin, kodlarının veya token'larının ve zamana dayalı tek kullanımlık parolaların (TOTP) belirli bir ömre sahip olduğu doğrulanmalıdır. Bant dışı istekler en fazla 10 dakika, TOTP'ler ise en fazla 30 saniye geçerli olmalıdır.	2
6.5.6	Herhangi bir kimlik doğrulama faktörünün (fiziksel cihazlar dahil) çalınması veya kaybı durumunda iptal edilebildiği doğrulanmalıdır.	3
6.5.7	Biyometrik kimlik doğrulama mekanizmalarının yalnızca ikincil faktör olarak, "sahip olduğunuz bir şey"veya "bildiğiniz bir şey"ile birlikte kullanıldığı doğrulanmalıdır.	3

#	Açıklama	Seviye
6.5.8	Zamana dayalı tek kullanımlık parolaların (TOTP) doğrulamasının, güvenilir bir hizmetten alınan zaman kaynağına dayanarak yapıldığı ve güvenilir olmayan veya istemci tarafından sağlanan zaman kaynağına dayanmadığı doğrulanmalıdır.	3

V6.6 Bant Dışı Kimlik Doğrulama Mekanizmaları

Bu mekanizmalar genellikle kimlik doğrulama sunucusunun, fiziksel bir cihazla güvenli bir ikinci kanal üzerinden iletişim kurmasını (örneğin mobil cihazlara gönderilen bildirimler) içerir. Bu tür mekanizmalar, "sahip olduğunuz bir şey"olarak değerlendirilir.

E-posta ve VOIP gibi güvenli olmayan bant dışı kimlik doğrulama yöntemlerine izin verilmemektedir. PSTN ve SMS ile yapılan kimlik doğrulama şu anda NIST tarafından "sınırlandırılmış"doğrulama mekanizmaları olarak değerlendirilmekte ve yerlerine zamana dayalı tek kullanımlık parolalar (TOTP) veya benzer kriptografik mekanizmaların tercih edilmesi önerilmektedir. NIST SP 800-63B § 5.1.3.3, telefon veya SMS kullanılması gerekiyorsa cihaz değişimi, SIM değişikliği, numara taşıma gibi anormal davranışların risklerinin ele alınmasını önermektedir. Bu ASVS bölümü bunu zorunlu kılmasa da, bu önlemlerin alınmaması Seviye 2'deki hassas bir uygulamada veya Seviye 3'teki bir uygulamada ciddi bir güvenlik riski olarak değerlendirilmelidir.

Ayrıca NIST, yakın zamanda anlık bildirimlerin kullanılmamasını öneren bir yönerge oluşturmuştur. Bu ASVS bölümü bu konuda bir zorunluluk getirmese de, "push bombing"risklerine karşı farkındalık önemlidir.

#	Açıklama	Seviye
6.6.1	Tek kullanımlık parolaların (OTP) telefon veya SMS ile Public Switched Telephone Network (PSTN) üzerinden iletildiği kimlik doğrulama mekanizmalarının, yalnızca telefon numarası önceden doğrulanmışsa, alternatif daha güçlü yöntemler (örneğin TOTP) de sunuluyorsa ve kullanıcıya güvenlik riskleri hakkında bilgi veriliyorsa sunulduğu doğrulanmalıdır. Seviye 3 uygulamalar için telefon ve SMS seçenekleri sunulmamalıdır.	2
6.6.2	Bant dışı kimlik doğrulama isteklerinin, kodlarının veya token'larının, oluşturuldukları özgün kimlik doğrulama isteğiyle bağlantılı olduğu ve daha önceki ya da sonraki bir istek için kullanılamadığı doğrulanmalıdır.	2
6.6.3	Kod tabanlı bant dışı kimlik doğrulama mekanizmalarının, oran sınırlaması ile kaba kuvvet saldırılarına karşı korunduğu doğrulanmalıdır. En az 64 bit entropiye sahip kodlar kullanılması da değerlendirilmelidir.	2

#	Açıklama	Seviye
6.6.4	Çok faktörlü kimlik doğrulama için push bildirimleri kullanılıyorsa, push bombing saldırılarına karşı oran sınırlamasının kullanıldığı doğrulanmalıdır. “Numara eşleştirme”de bu riski azaltabilir.	3

V6.7 Kriptografik Kimlik Doğrulama Mekanizması

Kriptografik kimlik doğrulama mekanizmaları, akıllı kartlar veya FIDO anahtarları gibi kullanıcıların kimlik doğrulama işlemini tamamlamak için kriptografik cihazı bilgisayara takması ya da eşleştirmesini gerektiren mekanizmaları içerir. Kimlik doğrulama sunucusu, kriptografik cihaza veya yazılıma bir “challenge nonce”gönderir; cihaz ya da yazılım da güvenli bir şekilde saklanan kriptografik anahtara dayalı olarak bir yanıt üretir. Bu bölümdeki gereksinimler bu mekanizmalar için, “Kriptografi”bölümünde ele alınan kriptografik algoritmalarla ilgili rehberlik sağlayan kısımlarla birlikte, uygulamaya özel rehberlik sağlar.

Kriptografik kimlik doğrulama için paylaşımlı veya gizli anahtarların kullanıldığı durumlarda, diğer sistem gizli anahtarlarıyla aynı mekanizmalar kullanılarak saklanmalıdır. Bu mekanizmalar “Yapılandırma”başlığındaki “Gizli Yönetimi”bölümünde belgelenmiştir.

Bu bölümdeki gereksinimler çoğunlukla NIST’s Guidance’ın § 5.1.7.2 kısmı ile ilişkilidir.

#	Açıklama	Seviye
6.7.1	Kriptografik kimlik doğrulama beyanlarını doğrulamak için kullanılan sertifikaların, değiştirilmeye karşı korumalı şekilde saklandığı doğrulanmalıdır.	3
6.7.2	Challenge nonce değerinin en az 64 bit uzunluğunda olduğu ve istatistiksel olarak benzersiz ya da cihazın ömrü boyunca benzersiz olduğu doğrulanmalıdır.	3

V6.8 Kimlik Sağlayıcısı ile Kimlik Doğrulama

Kimlik Sağlayıcılar (IdP’ler), kullanıcılar için birleştirilmiş kimlikler sunar. Kullanıcılar genellikle birden fazla IdP ile birden fazla kimliğe sahiptir; örneğin Azure AD, Okta, Ping Identity veya Google gibi bir kurumsal kimlik ya da Facebook, Twitter, Google veya WeChat gibi bir tüketici kimliği. Bu liste, bu şirketlere veya hizmetlere bir onay niteliğinde değil, sadece geliştiricilerin çoğu kullanıcının halihazırda birçok kimliğe sahip olduğu gerçeğini dikkate almaları gerektiğine dair bir hatırlatmadır. Kuruluşlar, IdP’nin kimlik doğrulama gücüne ilişkin risk profiline göre mevcut kullanıcı kimlikleriyle entegrasyon sağlamayı değerlendirmelidir. Örneğin, sahte veya geçici kimliklerin kolayca oluşturulabilmesi nedeniyle bir devlet kurumu, sosyal medya kimliklerini hassas sistemlere giriş

için kabul etmeyebilir; ancak bir mobil oyun şirketi, aktif oyuncu kitlesini artırmak için büyük sosyal medya platformlarıyla entegrasyon yapmayı tercih edebilir.

Harici kimlik sağlayıcıların güvenli kullanımı, kimlik sahtekarlığını veya sahte beyanları önlemek için dikkatli yapılandırma ve doğrulama gerektirir. Bu bölüm, bu riskleri ele almak için gereksinimleri sunar.

#	Açıklama	Seviye
6.8.1	Uygulama birden fazla kimlik sağlayıcıyı (IdP) destekliyorsa, başka bir desteklenen kimlik sağlayıcı aracılığıyla (örneğin aynı kullanıcı tanımlayıcısını kullanarak) kullanıcının kimliğinin taklit edilemediği doğrulanmalıdır. Standart önlem, uygulamanın kullanıcıyı IdP kimliği (bir ad alanı işlevi görür) ve IdP içindeki kullanıcı kimliğinin birleşimiyle kaydetmesi ve tanımlamasıdır.	2
6.8.2	Kimlik doğrulama beyanları üzerindeki dijital imzaların (örneğin JWT veya SAML beyanları) varlığı ve bütünlüğünün her zaman doğrulandığı ve imzasız veya geçersiz imzalı beyanların reddedildiği doğrulanmalıdır.	2
6.8.3	SAML beyanlarının benzersiz biçimde işlendiği ve yalnızca geçerlilik süresi içinde bir kez kullanıldığı, tekrar oynatma (replay) saldırılarını önlemek amacıyla doğrulanmalıdır.	2
6.8.4	Uygulama ayrı bir Kimlik Sağlayıcı (IdP) kullanıyorsa ve belirli işlemler için belirli kimlik doğrulama gücü, yöntemleri veya zaman bilgisi bekliyorsa, uygulamanın bu bilgileri IdP tarafından döndürülen veriler üzerinden doğruladığı teyit edilmelidir. Örneğin OIDC kullanılıyorsa, bu durum ID Token üzerindeki 'acr', 'amr'ye 'auth_time'gibi claim'lerin (varsa) doğrulanması ile sağlanabilir. Eğer IdP bu bilgileri sağlamıyorsa, uygulamanın asgari kimlik doğrulama gücünün kullanıldığını varsayan (örneğin yalnızca kullanıcı adı ve parola ile tek faktörlü doğrulama) belgelenmiş bir yedek yaklaşımı olmalıdır.	2

Referanslar

Daha fazla bilgi için:

- NIST SP 800-63 - Digital Identity Guidelines
- NIST SP 800-63B - Authentication and Lifecycle Management
- NIST SP 800-63 FAQ
- OWASP Testing Guide 4.0: Testing for Authentication
- OWASP Cheat Sheet - Password storage
- OWASP Cheat Sheet - Forgot password

- OWASP Cheat Sheet - Choosing and using security questions
- CISA Guidance on “Number Matching”
- Details on the FIDO Alliance

V7 Oturum Yönetimi

Kontrol Amacı

Oturum yönetim mekanizmaları, durum bilgisi olmayan iletişim protokolleri (örneğin HTTP) kullanılırken bile uygulamaların kullanıcı ve cihaz etkileşimlerini zaman içinde ilişkilendirmesini sağlar. Modern uygulamalar, farklı özelliklere ve amaçlara sahip birden fazla oturum belirteci kullanabilir. Güvenli bir oturum yönetim sistemi, saldırganların bir kurbanın oturumunu ele geçirmesini, kullanmasını veya kötüye kullanmasını önlemelidir. Oturumları yöneten uygulamaların aşağıdaki üst düzey oturum yönetimi gereksinimlerini karşılaması gerekir:

- Oturumlar her bireye özgüdür, tahmin edilemez veya paylaşılamaz.
- Oturumlar artık gerekmediğinde geçersiz kılınır ve inaktivite durumunda zaman aşımına uğrar.

Bu bölümdeki birçok gereksinim, yaygın tehditlere ve sıkça kötüye kullanılan kimlik doğrulama zayıflıklarına odaklanarak seçilmiş NIST SP 800-63 Digital Identity Guidelines kontrolleriyle ilgilidir.

Belirli oturum yönetim mekanizmalarının bazı uygulama detaylarına ilişkin gereksinimler diğer bölümlerde yer almaktadır:

- HTTP Çerezleri, oturum belirteçlerini güvence altına almak için yaygın bir mekanizmadır. Çerezler için özel güvenlik gereksinimleri “Web Frontend Güvenliği”bölümünde bulunabilir.
- Kendinden içerikli token’lar, oturumları sürdürmenin yaygın yollarından biridir. Bu token’lara özel güvenlik gereksinimleri “Kendinden İçerikli Token’lar”bölümünde ele alınmıştır.

V7.1 Oturum Yönetim Dokümantasyonu

Tüm uygulamalara uyan tek bir kalıp yoktur. Bu nedenle, tüm durumlara uygun evrensel sınırlar ve limitler tanımlamak mümkün değildir. Oturum yönetimi uygulama ve testine başlanmadan önce, oturum işleme ile ilgili güvenlik kararlarının belgelenmiş olduğu bir risk analizi yapılmalıdır. Bu, oturum yönetim sisteminin uygulamanın özel ihtiyaçlarına göre uyarlanmasını sağlar.

Durum bilgili ya da durumsuz (stateful ya da stateless) oturum mekanizması tercih edilse bile, analiz eksiksiz ve belgelenmiş olmalı ve seçilen çözümün ilgili tüm güvenlik gereksinimlerini karşılayabildiğini göstermelidir. Kullanılan herhangi bir Tek Oturum Açma (SSO) mekanizması ile olan etkileşim de değerlendirilmelidir.

#	Açıklama	Seviye
7.1.1	Kullanıcının oturum inaktivite zaman aşımı süresi ve mutlak maksimum oturum süresinin belgelendiği, diğer kontrollerle birlikte uygun olduğu ve NIST SP 800-63B yeniden kimlik doğrulama gereksinimlerinden herhangi bir sapma varsa gerekçesinin belgede yer aldığı doğrulanmalıdır.	2
7.1.2	Belgede bir hesap için kaç adet eşzamanlı (paralel) oturuma izin verildiği ile birlikte, maksimum aktif oturum sayısına ulaşıldığında uygulanacak davranış ve işlemlerin tanımlandığı doğrulanmalıdır.	2
7.1.3	Birleşik kimlik yönetim ekosisteminin bir parçası olarak kullanıcı oturumu oluşturan ve yöneten tüm sistemlerin, oturum ömrü, sonlandırma ve yeniden kimlik doğrulama gerektiren diğer koşulların nasıl koordine edildiğini açıklayan kontrollerle birlikte belgede yer aldığı doğrulanmalıdır.	2

V7.2 Temel Oturum Yönetimi Güvenliği

Bu bölüm, oturum token'larının güvenli bir şekilde oluşturulup doğrulandığını teyit eden, güvenli oturumların temel gereksinimlerini içerir.

#	Açıklama	Seviye
7.2.1	Uygulamanın tüm oturum token'ı doğrulama işlemlerini güvenilir bir backend servisi aracılığıyla gerçekleştirdiği doğrulanmalıdır.	1
7.2.2	Uygulamanın, statik API gizli anahtarları veya sabit değerler yerine, oturum yönetimi için dinamik olarak oluşturulan kendi kendine yeterli veya referans token'ları kullandığı doğrulanmalıdır.	1
7.2.3	Uygulama kullanıcı oturumlarını temsil etmek için referans token'ları kullanıyorsa, bu token'ların benzersiz olduğu, kriptografik olarak güvenli bir sözde rastgele sayı üreteci (CSPRNG) kullanılarak üretildiği ve en az 128 bit entropiye sahip olduğu doğrulanmalıdır.	1
7.2.4	Uygulamanın, kullanıcı kimlik doğrulaması (yeniden kimlik doğrulama dahil) sırasında yeni bir oturum token'ı oluşturduğu ve mevcut oturum token'ını sonlandırdığı doğrulanmalıdır.	1

V7.3 Oturum Zaman Aşımı

Oturum zaman aşımı mekanizmaları, oturum gaspı (session hijacking) ve diğer oturum kötüye kullanımı biçimlerine karşı fırsat aralığını en aza indirmeyi amaçlar. Zaman aşımı süreleri, belgelenmiş

güvenlik kararlarını karşılamalıdır.

#	Açıklama	Seviye
7.3.1	Risk analizi ve belgelenmiş güvenlik kararlarına göre yeniden kimlik doğrulamanın zorunlu kılınmasını sağlayacak şekilde bir hareketsizlik zaman aşımı bulunduğu doğrulanmalıdır.	2
7.3.2	Risk analizi ve belgelenmiş güvenlik kararlarına göre yeniden kimlik doğrulamanın zorunlu kılınmasını sağlayacak şekilde mutlak bir maksimum oturum süresi bulunduğu doğrulanmalıdır.	2

V7.4 Oturum Sonlandırma

Oturum sonlandırma işlemi, uygulamanın kendisi tarafından veya oturum yönetimi uygulama yerine SSO sağlayıcısı tarafından yürütülüyorsa SSO sağlayıcısı tarafından gerçekleştirilebilir. Bu bölümdeki gereksinimler değerlendirilirken SSO sağlayıcısının kapsam dahilinde olup olmadığına karar verilmesi gerekebilir çünkü bazı kontroller sağlayıcı tarafından yürütülüyor olabilir.

Oturum sonlandırma, yeniden kimlik doğrulamanın gerekli olmasını sağlamalı ve uygulama genelinde, birleşik oturum açma (varsa) ve tüm güvenen taraflar genelinde etkili olmalıdır.

Durum bilgili oturum mekanizmaları için sonlandırma tipik olarak oturumu backend tarafında geçersiz kılmayı içerir. Kendinden içerikli token'lar durumunda ise, bu token'ları geçersiz kılmak veya engellemek için ek önlemler gerekir; aksi halde bunlar süresi dolana kadar geçerli kalabilir.

#	Açıklama	Seviye
7.4.1	Oturum sonlandırma tetiklendiğinde (örneğin, çıkış yapıldığında veya süre dolduğunda), uygulamanın oturumun daha fazla kullanılmasına izin vermediği doğrulanmalıdır. Referans token'lar veya durum bilgili oturumlar için bu, oturum verilerinin uygulama arka ucunda geçersiz kılınması anlamına gelir. Kapsamlı token kullanan uygulamaların; sonlandırılmış token listesi tutmak, kullanıcıya özel tarih ve saatten önce oluşturulan token'ları reddetmek veya kullanıcıya özel imzalama anahtarını yenilemek gibi bir çözüm uygulaması gerekir.	1
7.4.2	Bir kullanıcı hesabı devre dışı bırakıldığında veya silindiğinde (örneğin bir çalışanın şirketten ayrılması), uygulamanın tüm aktif oturumları sonlandırdığı doğrulanmalıdır.	1

#	Açıklama	Seviye
7.4.3	Herhangi bir kimlik doğrulama faktörünün başarılı bir şekilde değiştirilmesi veya kaldırılmasından (şifre sıfırlama veya kurtarma yoluyla yapılan değişiklikler dahil ve varsa MFA ayarları güncellemeleri dahil) sonra, uygulamanın diğer tüm aktif oturumları sonlandırma seçeneği sunduğu doğrulanmalıdır.	2
7.4.4	Kimlik doğrulama gerektiren tüm sayfalarda kolay erişilebilir ve görünür bir çıkış yap (logout) işlevi bulunduğu doğrulanmalıdır.	2
7.4.5	Uygulama yöneticilerinin, bireysel bir kullanıcının veya tüm kullanıcıların aktif oturumlarını sonlandırabilme yetkisine sahip olduğu doğrulanmalıdır.	2

V7.5 Oturum Kötüye Kullanımına Karşı Savunmalar

Bu bölüm, aktif oturumların ele geçirilmesi veya kötüye kullanılması riskini azaltmaya yönelik gereksinimleri içerir. Bu riskler, örneğin kimliği doğrulanmış bir kurbanın internet tarayıcısının, kurbanın oturumunu kullanarak bir eylemi gerçekleştirmeye zorlanması gibi vektörler aracılığıyla ortaya çıkabilir.

Bu bölümdeki gereksinimler değerlendirilirken “Kimlik Doğrulama”bölümündeki seviyeye özel yönergeler dikkate alınmalıdır.

#	Açıklama	Seviye
7.5.1	E-posta adresi, telefon numarası, MFA yapılandırması veya hesap kurtarmada kullanılan diğer bilgiler gibi kimlik doğrulamayı etkileyebilecek hassas hesap özniteliklerinde değişiklik yapılmadan önce uygulamanın tam yeniden kimlik doğrulama gerektirdiği doğrulanmalıdır.	2
7.5.2	Kullanıcıların, (en az bir faktörle yeniden kimlik doğrulama yaptıktan sonra) mevcut tüm aktif oturumları görüntüleyebilmesi ve sonlandırabilmesinin mümkün olduğu doğrulanmalıdır.	2
7.5.3	Uygulamanın, son derece hassas işlemler veya operasyonlar gerçekleştirilmeden önce en az bir faktörle ek kimlik doğrulaması veya ikincil bir doğrulama gerektirdiği doğrulanmalıdır.	3

V7.6 Birleşik Yeniden Kimlik Doğrulama

Bu bölüm, Güvenen Taraf (Relying Party –RP) veya Kimlik Sağlayıcı (Identity Provider –IdP) kodu yazanlar için geçerlidir. Bu gereksinimler NIST SP 800-63C Federation & Assertions (Federasyon ve

Beyanlar) rehberinden türetilmiştir.

#	Açıklama	Seviye
7.6.1	Güvenen Taraf (RP) ile Kimlik Sağlayıcı (IdP) arasındaki oturum süresi ve oturum sonlandırma davranışlarının belgelenmiş olduğu ve, örneğin IdP kimlik doğrulama olayları arasında maksimum süreye ulaşıldığında yeniden kimlik doğrulamanın gerekli kılındığı doğrulanmalıdır.	2
7.6.2	Yeni bir oturum oluşturulmasının, kullanıcı onayı veya açık bir eylem gerektirdiği; kullanıcı etkileşimi olmaksızın yeni uygulama oturumlarının oluşturulmasının önlendiği doğrulanmalıdır.	2

Referanslar

Daha fazla bilgi için:

- OWASP Web Security Testing Guide: Session Management Testing
- OWASP Session Management Cheat Sheet

V8 Yetkilendirme

Kontrol Amacı

Yetkilendirme, yalnızca izin verilen tüketicilere (kullanıcılar, sunucular ve diğer istemciler) erişim verilmesini sağlar. En Az Ayrıcalık İlkesi'ni (Principle of Least Privilege - POLP) uygulamak için doğrulanmış uygulamalar aşağıdaki üst düzey gereksinimleri karşılamalıdır:

- Yetkilendirme kurallarını, karar alma faktörlerini ve çevresel bağlamları içerecek şekilde belgelendirin.
- Tüketiciler, yalnızca tanımlanmış haklarında izin verilen kaynaklara erişebilmelidir.

V8.1 Yetkilendirme Dokümantasyonu

Kapsamlı yetkilendirme dokümantasyonu, güvenlik kararlarının tutarlı biçimde uygulanmasını, denetlenebilir olmasını ve kurumsal politikalarla uyumlu olmasını sağlamak açısından kritik öneme sahiptir. Bu, geliştiriciler, yöneticiler ve test uzmanları için güvenlik gereksinimlerini açık ve uygulanabilir hâle getirerek yetkisiz erişim riskini azaltır.

#	Açıklama	Seviye
8.1.1	Yetkilendirme dokümantasyonunun, tüketici izinlerine ve kaynak özniteliklerine dayalı olarak işlev düzeyinde ve veriye özel erişim kısıtlamaları için kuralları tanımladığı doğrulanmalıdır.	1
8.1.2	Yetkilendirme dokümantasyonunun, tüketici izinlerine ve kaynak özniteliklerine dayalı olarak alan (field) düzeyinde erişim kısıtlamaları (hem okuma hem yazma) için kuralları tanımladığı doğrulanmalıdır. Bu kuralların, ilgili veri nesnesinin durum veya statü gibi diğer özniteliklerine bağlı olabileceği unutulmamalıdır.	2
8.1.3	Uygulamanın dokümantasyonunun, kimlik doğrulama ve yetkilendirme ile ilgili güvenlik kararlarının alınmasında kullanılan çevresel ve bağlamsal öznitelikleri (günün saati, kullanıcı konumu, IP adresi veya cihaz gibi) tanımladığı doğrulanmalıdır.	3
8.1.4	Kimlik doğrulama ve yetkilendirme dokümantasyonunun; işlev düzeyinde, veriye özel ve alan düzeyinde yetkilendirmenin yanı sıra çevresel ve bağlamsal faktörlerin karar almada nasıl kullanıldığını tanımladığı doğrulanmalıdır. Bu, değerlendirilen öznitelikleri, risk eşiklerini ve alınan aksiyonları (ör. izin ver, doğrulama iste, reddet, step-up kimlik doğrulama) içermelidir.	3

V8.2 Genel Yetkilendirme Tasarımı

İşlev, veri ve alan düzeyinde ayrıntılı yetkilendirme kontrollerinin uygulanması, tüketicilerin yalnızca kendilerine açıkça verilmiş izinlere sahip kaynaklara erişebilmesini sağlar.

#	Açıklama	Seviye
8.2.1	Uygulamanın, işlev düzeyindeki erişimin yalnızca açık izinlere sahip tüketicilerle sınırlandığını sağladığı doğrulanmalıdır.	1
8.2.2	Uygulamanın, veriye özel erişimin yalnızca belirli veri öğelerine açık izinlere sahip tüketicilerle sınırlandığını sağladığı doğrulanmalıdır. Bu, doğrudan nesne referansı güvenlik açıkları (IDOR) ve bozuk nesne düzeyinde yetkilendirme (BOLA) risklerini azaltır.	1
8.2.3	Uygulamanın, alan (field) düzeyindeki erişimin yalnızca belirli alanlara açık izinlere sahip tüketicilerle sınırlandığını sağladığı doğrulanmalıdır. Bu, bozuk nesne özelliği düzeyinde yetkilendirme (BOPLA) risklerini azaltır.	2

#	Açıklama	Seviye
8.2.4	Tüketicinin çevresel ve bağlamsal özniteliklerine (ör. günün saati, konum, IP adresi veya cihaz) dayalı uyarlanabilir güvenlik kontrollerinin, uygulama dokümantasyonunda tanımlandığı şekilde, kimlik doğrulama ve yetkilendirme kararları için uygulandığı doğrulanmalıdır. Bu kontroller, tüketici yeni bir oturum başlatmaya çalışırken ve mevcut bir oturum sırasında uygulanmalıdır.	3

V8.3 İşlem Düzeyinde Yetkilendirme

Yetkilendirme değişikliklerinin uygulamanın mimarisindeki uygun katmana anında yansıtılması, özellikle dinamik ortamlarda yetkisiz işlemleri önlemek açısından kritik öneme sahiptir.

#	Açıklama	Seviye
8.3.1	Uygulamanın yetkilendirme kurallarını güvenilir bir servis katmanında uyguladığı ve istemci taraflı JavaScript gibi güvenilmeyen tüketiciler tarafından değiştirilebilecek kontrollerle yetinmediği doğrulanmalıdır.	1
8.3.2	Yetkilendirme kararlarının temelini oluşturan değerlerdeki değişikliklerin hemen uygulandığı doğrulanmalıdır. Değişikliklerin anında uygulanamadığı durumlarda (örneğin self-contained token içindeki verilere güveniliyorsa), tüketici artık yetkili olmadığı hâlde bir işlem gerçekleştirdiğinde uyarı verecek ve değişikliği geri alacak önlemler uygulanmalıdır. Bu alternatifin bilgi sızıntısını önlemeyeceği unutulmamalıdır.	3
8.3.3	Bir nesneye erişimin, herhangi bir aracı veya onun adına hareket eden bir servis yerine yalnızca erişimi başlatan öznenin (ör. tüketici) izinlerine dayandığı doğrulanmalıdır. Örneğin, bir tüketici kimlik doğrulama için bir self-contained token kullanarak bir web servis çağırırsa ve bu servis başka bir servisten veri talep ederse, ikinci servis izin kararlarını verirken birinci servisin makine-makine token'ı yerine tüketicinin token'ını kullanmalıdır.	3

V8.4 Diğer Yetkilendirme Hususları

Yetkilendirme ile ilgili ek hususlar, özellikle yönetici arayüzleri ve çok kiracılı (multi-tenant) ortamlar için, yetkisiz erişimi önlemeye yardımcı olur.

#	Açıklama	Seviye
8.4.1	Multi-tenant uygulamaların, tüketici işlemlerinin izinli olmadıkları tenant' lar üzerinde asla etkili olmamasını sağlamak için tenant'lar arası kontroller kullandığı doğrulanmalıdır.	2
8.4.2	Yönetici arayüzlerine erişimin, sürekli tüketici kimlik doğrulaması, cihaz güvenlik durumu değerlendirmesi ve bağlamsal risk analizi dahil olmak üzere çok katmanlı güvenlik içerdiği doğrulanmalıdır. Yetkilendirme için yalnızca ağ konumu veya güvenilir uç noktalar yeterli olmamalıdır; ancak bu tür faktörler yetkisiz erişim olasılığını azaltabilir.	3

Referanslar

Daha fazla bilgi için:

- OWASP Web Security Testing Guide: Authorization
- OWASP Authorization Cheat Sheet

V9 Kendi İçinde Taşıyıcı Token (Self-contained Token)

Kontrol Amacı

Kendi içinde taşıyıcı token ya da self-contained token kavramı, ilk olarak 2012 tarihli RFC 6749 OAuth 2.0 standardında belirtilmiştir. Bu kavram, içeriğinde bir hizmetin güvenlik kararlarını almak için kullanacağı veri veya iddiaları (claim) taşıyan bir token'ı ifade eder. Bu, yalnızca bir tanımlayıcı içeren ve hizmetin veriyi yerel olarak aramasına yarayan basit token'lardan ayrıdır. Self-contained token'lara en yaygın örnekler JSON Web Tokens (JWT) ve SAML assertion'lardır.

Self-contained token kullanımı, OAuth ve OIDC dışında bile oldukça yaygın hale gelmiştir. Aynı zamanda bu mekanizmanın güvenliği, token'ın bütünlüğünün doğrulanabilmesine ve belirli bir bağlam için geçerli olduğunun teyit edilmesine bağlıdır. Bu süreçte birçok güvenlik açığı ortaya çıkabilir ve bu bölüm, uygulamaların alması gereken önlemleri ayrıntılı şekilde açıklamaktadır.

V9.1 Token Kaynağı ve Bütünlüğü

Bu bölüm, token'ın güvenilir bir tarafça üretildiğini ve üzerinde oynama yapılmadığını sağlamak için gereken gereksinimleri içerir.

#	Açıklama	Seviye
9.1.1	Self-contained token'ların içerikleri kabul edilmeden önce, oynama (tampering) saldırılarına karşı dijital imza veya MAC ile doğrulama yapıldığı doğrulanmalıdır.	1
9.1.2	Belirli bir bağlam için yalnızca bir allowlist üzerinde bulunan algoritmaların self-contained token oluşturmak ve doğrulamak için kullanılabildiği doğrulanmalıdır. Bu allowlist, yalnızca simetrik ya da yalnızca asimetrik algoritmaları içermeli ve kesinlikle 'None'algoritmasını içermemelidir. Hem simetrik hem de asimetrik algoritmalar desteklenmek zorundaysa, anahtar karışıklığını (key confusion) önlemek için ek kontroller uygulanmalıdır.	1
9.1.3	Self-contained token'ları doğrulamak için kullanılan anahtar malzemesinin, token issuer için önceden yapılandırılmış güvenilir kaynaklardan alındığı doğrulanmalıdır. Bu, saldırganların güvenilmeyen kaynak ve anahtarları belirtmesini engeller. JWT ve diğer JWS yapılarında 'jku', 'x5u've 'jwk'gibi header'lar güvenilir kaynaklardan oluşan bir allowlist'e göre doğrulanmalıdır.	1

V9.2 Token İçeriği

Bir self-contained token içeriğine dayanarak güvenlik kararı alınmadan önce, token'ın geçerlilik süresi içinde sunulup sunulmadığı ve belirli bir hizmet tarafından belirli bir amaç için kullanıma uygun olup olmadığı doğrulanmalıdır. Bu, aynı issuer'dan gelen farklı token türlerinin farklı hizmetler arasında güvensiz şekilde yeniden kullanılmasını önlemeye yardımcı olur.

OAuth ve OIDC'ye özgü gereksinimler ilgili bölümde yer almaktadır.

#	Açıklama	Seviye
9.2.1	Token verisinde bir geçerlilik zaman aralığı (validity time span) varsa, yalnızca doğrulama zamanı bu zaman aralığı içerisindeyse token ve içeriği kabul edilmelidir. Örneğin, JWT'ler için 'nbf've 'exp'claim'leri doğrulanmalıdır.	1
9.2.2	Token'ı alan hizmetin, token içeriğini kabul etmeden önce bu token'ın doğru türde olduğunu ve sunulma amacına uygun olduğunu doğruladığı teyit edilmelidir. Örneğin, yalnızca erişim token'ları yetkilendirme kararları için, yalnızca ID Token'lar kullanıcı kimlik doğrulamasını kanıtlamak için kullanılabilir.	2

#	Açıklama	Seviye
9.2.3	Hizmetin yalnızca kendisiyle kullanılması amaçlanan (audience) token'ları kabul ettiği doğrulanmalıdır. JWT'lerde bu, 'aud'claim'inin hizmet içinde tanımlı bir allowlist ile doğrulanması yoluyla sağlanabilir.	2
9.2.4	Eğer bir token issuer, farklı audience'lara token oluşturmak için aynı private key'i kullanıyorsa, oluşturulan token'ların hedef audience'ları benzersiz şekilde tanımlayan bir audience kısıtlaması içerdiği doğrulanmalıdır. Bu, bir token'ın farklı bir audience ile yeniden kullanılmasını önler. Audience tanımlayıcısı dinamik olarak sağlanıyorsa, token issuer bu audience'ları doğrulamalı ve audience taklitlerinin (impersonation) önüne geçmelidir.	2

Referanslar

Daha fazla bilgi için:

- OWASP JSON Web Token Cheat Sheet for Java Cheat Sheet (genel olarak yönlendirme için kullanışlı)

V10 OAuth ve OIDC

Kontrol Amacı

OAuth2 (bu bölümde kısaca OAuth olarak anılacaktır), yetki devri için endüstri standardı bir çerçevedir. Örneğin, OAuth kullanılarak bir istemci uygulama, kullanıcı tarafından yetkilendirildiği sürece, o kullanıcı adına API'lere (sunucu kaynaklarına) erişim sağlayabilir.

OAuth tek başına kullanıcı kimlik doğrulaması amacıyla tasarlanmamıştır. OpenID Connect (OIDC) çerçevesi, OAuth üzerine bir kullanıcı kimliği katmanı ekleyerek bu eksikliği tamamlar. OIDC, standartlaştırılmış kullanıcı bilgileri, Single Sign-On (SSO) ve oturum yönetimi gibi özellikleri destekler. OIDC, OAuth'un bir uzantısı olduğundan, bu bölümdeki OAuth gereksinimleri aynı zamanda OIDC için de geçerlidir.

OAuth çerçevesinde aşağıdaki roller tanımlanır:

- OAuth istemcisi, sunucu kaynaklarına erişim sağlamaya çalışan uygulamadır (örneğin, verilen access token'ı kullanarak bir API çağrısı yapar). Genellikle bu, sunucu taraflı bir uygulamadır.
  - Gizli istemci (confidential client), authorization server ile kimlik doğrulama sırasında kullandığı kimlik bilgilerini gizli tutabilecek kapasitede olan istemcidir.
  - Genel istemci (public client), authorization server ile kimlik doğrulama sırasında kullandığı kimlik bilgilerini gizli tutamayacak durumdadır. Bu nedenle kendini yalnızca bir client_id ile tanıtır, client_id ve client_secret gibi bilgilerle kimlik doğrulaması yapmaz.

- OAuth kaynak sunucusu (resource server –RS), OAuth istemcilerine kaynakları sunan API'dir.
- OAuth yetkilendirme sunucusu (authorization server –AS), access token üreten sunucu uygulamasıdır. Bu token'lar sayesinde OAuth istemcisi, son kullanıcı adına veya kendi adına RS üzerindeki kaynaklara erişim sağlayabilir. AS genellikle ayrı bir uygulama olsa da, uygun görüldüğünde RS ile entegre olabilir.
- Kaynak sahibi (resource owner –RO), OAuth istemcilerinin kendi adına, kaynak sunucusu üzerinde kısıtlı erişim elde etmesine izin veren son kullanıcıdır. Yetkilendirme sürecine AS üzerinden katılım sağlar.

OIDC çerçevesinde aşağıdaki roller tanımlanır:

- Güvenen taraf (relying party –RP), son kullanıcının kimlik doğrulamasını OpenID Provider üzerinden gerçekleştirmek isteyen istemci uygulamadır. Aynı zamanda bir OAuth istemcisidir.
- OpenID Provider (OP), son kullanıcıyı kimlik doğrulama yeteneğine sahip olan ve RP'ye OIDC claim'lerini sağlayan bir OAuth yetkilendirme sunucusudur. OP çoğu zaman bir kimlik sağlayıcı (IdP) olur; ancak federasyon senaryolarında OP ve gerçek kimlik sağlayıcı farklı uygulamalar olabilir.

OAuth ve OIDC başlangıçta üçüncü taraf uygulamalar için tasarlanmıştı. Ancak günümüzde ilk taraf uygulamalar tarafından da yaygın şekilde kullanılmaktadır. Bu kullanım senaryolarında, özellikle kimlik doğrulama ve oturum yönetimi amacıyla kullanıldığında protokol belirli bir karmaşıklık ekler ve bu durum bazı yeni güvenlik riskleri yaratabilir.

OAuth ve OIDC birçok farklı uygulama tipiyle kullanılabilir; ancak ASVS ve bu bölümdeki gereksinimler özellikle web uygulamaları ve API'ler için tasarlanmıştır.

OAuth ve OIDC, web teknolojilerinin üzerine kurulu mantıksal protokoller olduklarından, bu bölümdeki gereksinimler diğer bölümlerden bağımsız değerlendirilemez; genel güvenlik gereksinimleri geçerliliğini korur.

Bu bölüm, https://oauth.net/2/ ve https://openid.net/developers/specs/ üzerindeki spesifikasyonlarla uyumlu şekilde OAuth2 ve OIDC için güncel en iyi uygulamaları ele alır. RFC'ler olgun belgeler olarak görülse de, sık sık güncellenmektedir. Bu nedenle uygulamada, ilgili gereksinimlerin en güncel versiyonlarla uyumlu şekilde ele alınması önemlidir. Ayrıntılar için bölümün referans kısmına bakınız.

Bu alanın karmaşıklığı göz önünde bulundurulduğunda, güvenli bir OAuth veya OIDC çözümünün, tanınmış ve endüstri standardı yetkilendirme sunucularını kullanması ve önerilen güvenlik yapılandırmalarını uygulaması son derece önemlidir.

Bu bölümde kullanılan terimler, OAuth ve OIDC spesifikasyonlarındaki tanımlarla uyumludur. Ancak OIDC'ye özel gereksinimler dışında, genellikle OAuth terminolojisi tercih edilmiştir.

Bu bölümde geçen "token"terimi, aşağıdaki anlamlarda kullanılmıştır:

- Access token'ları sadece RS tarafından kullanılır. Bunlar ya introspection ile doğrulanan bir reference token olabilir ya da belirli bir anahtar malzemesiyle doğrulanan bir self-contained token olabilir.

- Refresh token'ları, yalnızca onu üreten authorization server tarafından kullanılabilir.
- OIDC ID Token'ları sadece authorization flow'u tetikleyen istemci tarafından kullanılabilir.

Bu bölümdeki bazı gereksinimlerin risk seviyesi, istemcinin gizli istemci mi yoksa açık istemci mi olduğuna göre değişiklik gösterebilir. Güçlü istemci kimlik doğrulaması, birçok saldırı vektörünü etkisiz hale getirebildiği için, L1 uygulamalarında bazı gereksinimler gizli istemci kullanıldığında esnetilebilir.

V10.1 Genel OAuth ve OIDC Güvenliği

Bu bölüm, OAuth veya OIDC kullanan tüm uygulamalara uygulanabilen genel mimari gereksinimleri kapsar.

#	Açıklama	Seviye
10.1.1	Token'ların yalnızca kesinlikle ihtiyaç duyan bileşenlere gönderildiği doğrulanmalıdır. Örneğin, tarayıcı tabanlı JavaScript uygulamaları için bir backend-for-frontend modeli kullanıldığında, access ve refresh token'lara yalnızca backend'in erişebilmesi gerekir.	2
10.1.2	Authorization server'dan gelen değerler (örneğin authorization code veya ID Token gibi) yalnızca aynı user agent oturumu ve işlemi tarafından başlatılmış bir yetkilendirme akışının sonucuysa kabul edildiği doğrulanmalıdır. Bu, client tarafından oluşturulan secret'ların 'proof key for code exchange' (PKCE) 'code_verifier', 'state'veya OIDC 'nonce'gibi tahmin edilemez, işleme özgü ve hem client'a hem de işlem başlatılan user agent oturumuna güvenli bir şekilde bağlı olmasını gerektirir.	2

V10.2 OAuth İstemcisi

Bu gereksinimler, OAuth istemci uygulamaları için sorumlulukları detaylandırır. Client örneğin bir web sunucusu backend'i (genellikle Backend For Frontend olarak hareket eden), bir backend servis entegrasyonu veya bir frontend Single Page Application (SPA, yani tarayıcı tabanlı uygulama) olabilir.

Genel olarak backend istemciler gizli istemci olarak, frontend istemciler ise açık istemci olarak değerlendirilir. Ancak, son kullanıcı cihazında çalışan native uygulamalar, 'OAuth dynamic client registration'kullanıldığında gizli olarak kabul edilebilir.

#	Açıklama	Seviye
10.2.1	Code flow kullanılıyorsa, OAuth client'ın token taleplerini tetikleyen CSRF saldırılarına karşı korumaya sahip olduğu doğrulanmalıdır. Bu genellikle PKCE kullanılarak veya authorization isteğinde gönderilen 'state' parametresinin kontrolü ile sağlanır.	2
10.2.2	OAuth istemcisi birden fazla AS ile etkileşime girebiliyorsa, mix-up saldırılarına karşı korumaya sahip olduğu doğrulanmalıdır. Örneğin, authorization server'ın 'iss'parametresini döndürmesi zorunlu kılınabilir ve bu parametrenin hem authorization yanıtında hem de token yanıtında doğrulanması gerekir.	2
10.2.3	OAuth istemcisinin authorization server'a yaptığı taleplerde yalnızca gerekli scope'ları (veya diğer yetkilendirme parametrelerini) talep ettiği doğrulanmalıdır.	3

V10.3 OAuth Kaynak Sunucusu (RS)

ASVS ve bu bölüm bağlamında resource server (RS) bir API'dir. Güvenli erişim sağlamak için RS şunları yapmalıdır:

- Access token'ı, token formatı ve ilgili protokol spesifikasyonlarına göre doğrulamalıdır; örneğin JWT doğrulaması veya OAuth token introspection.
- Geçerliyse, access token'dan alınan bilgiler ve verilmiş izinlere göre yetkilendirme kararları uygulamalıdır. Örneğin, resource server'ın client'ın (RO adına hareket eden) istenen kaynağa erişme yetkisine sahip olduğunu doğrulaması gerekir.

Bu nedenle burada listelenen gereksinimler, OAuth veya OIDC'ye özeldir ve token doğrulaması yapıldıktan sonra, token içeriğine dayalı yetkilendirme gerçekleştirilmeden önce uygulanmalıdır.

#	Açıklama	Seviye
10.3.1	RS'nin yalnızca bu servisle kullanılmak üzere tasarlanmış access token'ları kabul ettiği doğrulanmalıdır (audience kontrolü). Audience bilgisi yapılandırılmış token'larda (örneğin JWT içindeki 'aud'claim'i) yer alabilir veya token introspection endpoint aracılığıyla kontrol edilebilir.	2
10.3.2	RS'nin, access token'dan gelen claim'lere dayanarak yetki devri tanımına uygun şekilde yetkilendirme kararları uyguladığı doğrulanmalıdır. 'sub', 'scope've 'authorization_details'gibi claim'ler varsa, bu kararın parçası olmalıdır.	2

#	Açıklama	Seviye
10.3.3	Erişim kontrolü kararında, access token'dan (JWT veya ilgili token introspection yanıtı) benzersiz bir kullanıcının tanımlanması gerekiyorsa, RS'nin başka kullanıcılara devredilemeyecek claim'ler ile kullanıcıyı tanımladığı doğrulanmalıdır. Genellikle bu, 'iss'ye 'sub'claim'lerinin birlikte kullanılması anlamına gelir.	2
10.3.4	RS'nin belirli bir kimlik doğrulama gücü, yöntemi veya güncelliği (recentness) gerektirmesi durumunda, sunulan access token'ın bu kısıtlamaları karşıladığını doğruladığı, örneğin, OIDC 'acr', 'amr'ye 'auth_time'claim'leri kullanılarak doğrulanmalıdır.	2
10.3.5	RS'nin, (yetkisiz taraflarca) çalınmış veya tekrar edilen access token'ların kullanımını önlemek için sender-constrained access token'lar kullandığı doğrulanmalıdır. Bu, OAuth 2 için Mutual TLS veya OAuth 2 Demonstration of Proof of Possession (DPoP) ile sağlanabilir.	3

V10.4 OAuth Yetkilendirme Sunucusu (AS)

Bu gereksinimler, OAuth authorization server'ların (yetkilendirme sunucuları) ve OpenID Provider'ların sorumluluklarını detaylandırır.

İstemci kimlik doğrulaması için 'self_signed_tls_client_auth'yöntemi, RFC 8705 böüm 2.2'de belirtilen ön koşullarla birlikte kullanılabilir.

#	Açıklama	Seviye
10.4.1	AS'nin, istemciye özel önceden kaydedilmiş URI'lerden oluşan bir allowlist kullanarak redirect URI'leri tam dize karşılaştırması ile doğruladığı doğrulanmalıdır.	1
10.4.2	AS'nin, "authorization response"içinde "authorization code"döndürdüğü durumlarda, bu kodun yalnızca bir kez kullanılabildiği doğrulanmalıdır. Daha önce access token vermek için kullanılmış bir authorization code ile yapılan ikinci geçerli token isteği reddedilmeli ve ilgili tüm token'lar iptal edilmelidir.	1
10.4.3	Authorization code'un kısa ömürlü olduğu doğrulanmalıdır. L1 ve L2 uygulamalar için maksimum yaşam süresi 10 dakika, L3 uygulamalar için 1 dakikadır.	1

#	Açıklama	Seviye
10.4.4	Belirli bir istemci için AS'nin yalnızca o istemcinin ihtiyaç duyduğu grant türlerine izin verdiği doğrulanmalıdır. "Token"(Implicit flow) ve "password" (Resource Owner Password Credentials flow) grant türlerinin artık kullanılmaması gerekir.	1
10.4.5	AS'nin, açık istemciler için refresh token replay saldırılarını azalttığı doğrulanmalıdır. Tercihen sender-constrained refresh token'lar (DPoP veya mutual TLS ile certificate-bound access token'lar) kullanılmalıdır. L1 ve L2 uygulamalar için refresh token rotation kullanılabilir. Rotation kullanılıyorsa, AS refresh token'ı kullanım sonrası geçersiz kılmalı ve geçersiz bir refresh token sunulması durumunda o yetkilendirme ile ilişkili tüm refresh token'ları iptal etmelidir.	1
10.4.6	Code grant kullanılıyorsa, AS'nin authorization code interception saldırılarına karşı koruma sağladığı doğrulanmalıdır. Yetkilendirme isteğinde geçerli bir "code_challenge"değeri zorunlu olmalı ve "code_challenge_method"olarak "plain"kabul edilmemelidir. Token isteğinde ise "code_verifier"parametresi doğrulanmalıdır.	2
10.4.7	AS'nin, kimlik doğrulaması yapılmamış dinamik istemci kaydını desteklemesi durumunda, kötü niyetli istemci uygulamalarına karşı riskleri azalttığı doğrulanmalıdır. Kayıtlı URI'ler gibi client metadata'sı doğrulanmalı, kullanıcı onayı alınmalı ve güvenilmeyen bir client uygulamasıyla yapılan yetkilendirme isteği işlenmeden önce kullanıcı uyarılmalıdır.	2
10.4.8	Refresh token'ların, sliding expiration kullanılsa bile mutlak bir sona erme süresine sahip olduğu doğrulanmalıdır.	2
10.4.9	Yetkili bir kullanıcı tarafından AS kullanıcı arayüzü üzerinden refresh token' ların ve reference access token'ların iptal edilebildiği doğrulanmalıdır.	2
10.4.10	Gizli istemcilerin token istekleri, pushed authorization request (PAR) ve token iptal istekleri gibi server'a yapılan backchannel isteklerinde kimliğinin kontrol edildiği doğrulanmalıdır.	2
10.4.11	AS yapılandırmasının, OAuth istemcisine yalnızca gerekli scope'ları atadığı doğrulanmalıdır.	2
10.4.12	Belirli bir client için, authorization server'ın yalnızca client'ın ihtiyaç duyduğu 'response_mode'değerine izin verdiği doğrulanmalıdır. Örneğin, bu değer ya beklenenlerle karşılaştırılarak doğrulanmalı ya da PAR veya JAR kullanılmalıdır.	3

#	Açıklama	Seviye
10.4.13	“Code”grant türünün daima pushed authorization request (PAR) ile birlikte kullanıldığı doğrulanmalıdır.	3
10.4.14	AS’nin yalnızca sender-constrained (Proof-of-Possession) access token’lar ürettiği doğrulanmalıdır. Bu, mutual TLS ile certificate-bound access token ya da DPoP ile DPoP-bound access token olabilir.	3
10.4.15	Sunucu taraflı istemciler için (kullanıcı cihazında çalışmayan), AS’nin “authorization_details”parametresinin istemci backend’i tarafından sağlandığını ve kullanıcı tarafından değiştirilmediğini doğruladığı, örneğin pushed authorization request (PAR) or JWT-secured Authorization Request (JAR) zorunlu kılınarak doğrulanmalıdır.	3
10.4.16	Bir istemci gizli ise, AS’nin güçlü istemci kimlik doğrulama yöntemlerini zorunlu kıldığı doğrulanmalıdır. Bu yöntemler, public-key cryptography’ye dayalı ve replay saldırılarına karşı dayanıklı olmalı, örneğin mutual TLS (“tls_client_auth”, “self_signed_tls_client_auth”) veya private key JWT (“private_key_jwt”) gibi.	3

V10.5 OIDC İstemcisi

“OIDC relying party”, bir OAuth istemcisi olarak davrandığı için “OAuth İstemcisi”bölümündeki gereksinimler burada da geçerlidir.

“Kimlik Doğrulama”başlığındaki “Kimlik Sağlayıcısı ile Kimlik Doğrulama”bölümü de ilgili genel gereksinimleri içerir.

#	Açıklama	Seviye
10.5.1	İstemcinin (relying party olarak) ID Token replay saldırılarını azalttığı doğrulanmalıdır. Örneğin, ID Token’daki ‘nonce’claim’inin, OpenID Provider’a gönderilen kimlik doğrulama isteğindeki ‘nonce’değeriyle eşleştiği doğrulanmalıdır.	2
10.5.2	İstemcinin kullanıcıyı ID Token claim’lerinden (genellikle, başka kullanıcılarla yeniden atanamayan ‘sub’claim’i) benzersiz şekilde tanımladığı doğrulanmalıdır.	2
10.5.3	İstemcinin, kötü niyetli bir AS’nin başka bir AS’yi taklit etmesini AS metadata’sı üzerinden reddettiği doğrulanmalıdır. AS metadata’sındaki issuer URL’si, istemci tarafından beklenen önceden yapılandırılmış issuer URL’siyle tam olarak eşleşmiyorsa, metadata reddedilmelidir.	2

#	Açıklama	Seviye
10.5.4	İstemcinin, ID Token'ın kendisi için (audience) kullanılması amaçlandığını doğruladığı doğrulanmalıdır. Bu, token'daki 'aud'claim'inin istemcinin 'client_id'değeriyle eşit olup olmadığının kontrol edilmesiyle yapılır.	2
10.5.5	OIDC back-channel logout kullanıldığında, relying party'nin "forced logout" ve "logout"akışında "cross-JWT confusion"gibi saldırılara karşı koruma sağladığı doğrulanmalıdır. Client, logout token'ın 'logout+jwt'türünde olduğunu, 'event'claim'inin doğru üyeye sahip olduğunu ve 'nonce'claim'i içermediğini doğrulamalıdır. Ayrıca token için kısa bir sona erme süresi (örneğin 2 dakika) önerilir.	2

V10.6 OpenID Provider

OpenID Provider'lar, OAuth AS olarak davrandığı için "OAuth Yetkilendirme Sunucusu"bölümündeki gereksinimler burada da geçerlidir.

ID Token flow (code flow değil) kullanılıyorsa, access token üretilmez ve OAuth AS için geçerli olan birçok gereksinim uygulanamaz.

#	Açıklama	Seviye
10.6.1	OpenID Provider'ın yalnızca 'code', 'ciba', 'id_token'veya 'id_token code' response mode değerlerine izin verdiği doğrulanmalıdır. 'code'değeri, 'id_token code'(OIDC Hybrid flow) yerine tercih edilmelidir. 'token'(Implicit flow) kullanılmamalıdır.	2
10.6.2	OpenID Provider'ın forced logout yoluyla yapılan hizmet reddi saldırılarını önlediği doğrulanmalıdır. Bu, son kullanıcıdan açık onay alınarak veya varsa logout isteğinde (relying party tarafından başlatılan) bulunan parametrelerin, örneğin 'id_token_hint', doğrulanmasıyla yapılabilir.	2

V10.7 Onay (Consent) Yönetimi

Bu gereksinimler, kullanıcı onayının AS tarafından doğrulanmasını kapsar. Uygun kullanıcı onayı olmadan, kötü niyetli bir aktör spoofing veya sosyal mühendislik yoluyla kullanıcının adına yetki elde edebilir.

#	Açıklama	Seviye
10.7.1	AS'nin her yetkilendirme isteği için kullanıcının onayını aldığı doğrulanmalıdır. İstemcinin kimliği garanti altına alınamıyorsa, AS daima kullanıcıdan açıkça onay istemelidir.	2
10.7.2	AS'nin kullanıcıdan onay istediğinde, neye onay verildiğini açık ve yeterli şekilde gösterdiği doğrulanmalıdır. Uygun olduğunda bu bilgiler, istenen yetkilendirmelerin doğasını (genellikle scope, resource server, RAR gibi detaylara dayalı), yetkilendirilmiş uygulamanın kimliğini ve bu yetkilendirmelerin süresini içermelidir.	2
10.7.3	Kullanıcının, AS üzerinden verdiği onayları görüntüleyebildiği, değiştirebildiği ve iptal edebildiği doğrulanmalıdır.	2

Referanslar

OAuth ile ilgili daha fazla bilgi için:

- oauth.net
- OWASP OAuth 2.0 Protocol Cheat Sheet

ASVS içindeki OAuth ile ilgili gereksinimler için, yayımlanmış ve taslak durumundaki aşağıdaki RFC'ler kullanılmaktadır:

- RFC6749 The OAuth 2.0 Authorization Framework
- RFC6750 The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC6819 OAuth 2.0 Threat Model and Security Considerations
- RFC7636 Proof Key for Code Exchange by OAuth Public Clients
- RFC7591 OAuth 2.0 Dynamic Client Registration Protocol
- RFC8628 OAuth 2.0 Device Authorization Grant
- RFC8707 Resource Indicators for OAuth 2.0
- RFC9068 JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens
- RFC9126 OAuth 2.0 Pushed Authorization Requests
- RFC9207 OAuth 2.0 Authorization Server Issuer Identification
- RFC9396 OAuth 2.0 Rich Authorization Requests
- RFC9449 OAuth 2.0 Demonstrating Proof of Possession (DPoP)
- RFC9700 Best Current Practice for OAuth 2.0 Security
- draft OAuth 2.0 for Browser-Based Applications
- draft The OAuth 2.1 Authorization Framework

OpenID Connect ile ilgili daha fazla bilgi için:

- OpenID Connect Core 1.0

- FAPI 2.0 Security Profile

V11 Kriptografi

Kontrol Amacı

Bu bölümün amacı, kriptografinin genel kullanımı için en iyi uygulamaları tanımlamak, kriptografik ilkelere dair temel bir anlayış kazandırmak ve daha dayanıklı ve modern yaklaşımlara yönelimi teşvik etmektir. Bu doğrultuda aşağıdakileri teşvik eder:

- Güvenli şekilde hata veren, gelişen tehditlere uyum sağlayan ve geleceğe yönelik sağlam kriptografik sistemlerin uygulanması,
- Hem güvenli hem de sektörün en iyi uygulamalarıyla uyumlu kriptografik mekanizmaların kullanılması,
- Uygun erişim kontrolleri ve denetimle birlikte güvenli bir kriptografik anahtar yönetim sisteminin sürdürülmesi,
- Kriptografik ortamın düzenli olarak değerlendirilerek yeni risklerin analiz edilmesi ve algoritmaların buna göre uyarlanması,
- Uygulamanın yaşam döngüsü boyunca tüm kriptografik varlıkların keşfedildiğinden ve güvence altına alındığından emin olmak için kriptografi kullanım alanlarının belirlenmesi ve yönetilmesi.

Genel ilkeleri ve en iyi uygulamaları özetlemenin yanı sıra, bu belge Ek C - Kriptografi Standartları bölümünde gereksinimlerle ilgili daha teknik bilgiler de sunar. Bu bölüm, “onaylı”kabul edilen algoritmaları ve modları içerir.

Kriptografinin sır yönetimi ya da iletişim güvenliği gibi başka bir sorunu çözmek için kullanıldığı gereksinimler, bu standardın başka bölümlerinde yer almaktadır.

V11.1 Kriptografik Envanter ve Dokümantasyon

Veri varlıklarını sınıflandırmalarına uygun şekilde korumak için uygulamaların güçlü bir kriptografik mimari ile tasarlanması gerekir. Her şeyi şifrelemek kaynak israfına, hiçbir şeyi şifrelememek ise hukuki ihlallere yol açar. Genellikle mimari tasarım, tasarım sprintleri veya mimari zirveler sırasında bir denge kurulmalıdır. Kriptografiyi “doğaçlama”yapmak veya sonradan eklemek, güvenli biçimde uygulamak için her zaman daha maliyetli olacaktır.

Tüm kriptografik varlıkların düzenli olarak keşfedildiğinden, envantere alındığından ve değerlendirildiğinden emin olunması önemlidir. Bunun nasıl yapılacağı hakkında daha fazla bilgi için ekteki bölüme bakınız.

Kriptografik sistemlerin kuantum bilgisayarların olası yükselişine karşı geleceğe yönelik olarak dayanıklı hale getirilmesi de kritik önemdedir. Post-Kuantum Kriptografi (PQC), kuantum bilgisayar saldırılarına karşı güvenli kalacak şekilde tasarlanmış kriptografik algoritmaları ifade eder;

bu tür bilgisayarların, RSA ve eliptik eğri kriptografisi (ECC) gibi yaygın algoritmaları kırması beklenmektedir.

Onaylı PQC ilkelikleri ve standartlarıyla ilgili güncel bilgiler için ekteki bölüme bakınız.

#	Açıklama	Seviye
11.1.1	Kriptografik anahtarların yönetimi için belgelenmiş bir politika ve NIST SP 800-57 gibi bir anahtar yönetim standardını izleyen bir kriptografik anahtar yaşam döngüsü tanımlandığı doğrulanmalıdır. Bu, anahtarların aşırı miktarda paylaşılmadığını (örneğin, paylaşılan sırlar için ikiden fazla tarafla veya özel anahtarlar için birden fazla tarafla) garanti altına almalıdır.	2
11.1.2	Tüm kriptografik anahtarlar, algoritmalar ve uygulama tarafından kullanılan sertifikaları içeren bir kriptografik envanterin oluşturulduğu, sürdürüldüğü ve düzenli olarak güncellendiği doğrulanmalıdır. Bu envanter ayrıca, sistemde anahtarların nerede kullanılabileceğini ve kullanılamayacağını, ve hangi veri türlerinin bu anahtarlarla korunup korunamayacağını da belgelemelidir.	2
11.1.3	Sistem içerisinde şifreleme, özetleme ve imzalama işlemleri dahil olmak üzere kriptografi kullanımının tüm örneklerini tespit etmek için kriptografik keşif mekanizmalarının kullanıldığı doğrulanmalıdır.	3
11.1.4	Kriptografik bir envanterin sürdürüldüğü doğrulanmalıdır. Bu envanter, gelecekteki tehditlere karşı tepki verebilmek amacıyla post-kuantum kriptografiye geçiş yolunu belirten belgelenmiş bir plan içermelidir.	3

V11.2 Güvenli Kriptografi Uygulaması

Bu bölüm, bir uygulama için temel kriptografik algoritmaların seçimi, uygulanması ve sürekli yönetimine yönelik gereksinimleri tanımlar. Amaç, yalnızca sağlam, sektör tarafından kabul görmüş kriptografik ilkeliklerin, güncel standartlara (örneğin NIST, ISO/IEC) ve en iyi uygulamalara uygun biçimde kullanılmasını sağlamaktır. Kuruluşlar, her kriptografik bileşenin peer-review uygulanmış kanıtlar ve pratik güvenlik testlerine dayalı olarak seçildiğinden emin olmalıdır.

#	Açıklama	Seviye
11.2.1	Kriptografik işlemler için sektör tarafından doğrulanmış uygulamaların (kütüphaneler ve donanım hızlandırmalı uygulamalar dahil) kullanıldığı doğrulanmalıdır.	2

#	Açıklama	Seviye
11.2.2	Uygulamanın, rastgele sayı üreticileri, kimliği doğrulanmış şifreleme, MAC veya özetleme algoritmaları, anahtar uzunlukları, turlar, şifreler ve modların herhangi bir zamanda yeniden yapılandırılabilir, yükseltilebilir veya değiştirilebilir olacağı şekilde kripto çevikliği (crypto agility) ile tasarlandığı doğrulanmalıdır. Aynı şekilde, anahtarların ve parolaların değiştirilebilmesi ve verilerin yeniden şifrelenebilmesi mümkün olmalıdır. Bu, onaylanmış PQC şemalarının yüksek güvenlikli uygulamaları yaygın olarak kullanılabilir hale geldiğinde sorunsuz geçişe olanak tanır.	2
11.2.3	Tüm kriptografik ilkeliklerin, algoritma, anahtar boyutu ve yapılandırmaya göre minimum 128 bit güvenlik sağladığı doğrulanmalıdır. Örneğin, 256-bit ECC anahtarı yaklaşık 128 bit güvenlik sağlarken, RSA için aynı güvenliği elde etmek 3072-bit anahtar gerektirir.	2
11.2.4	Tüm kriptografik işlemlerin sabit süreli (constant-time) olduğu, karşılaştırmalarda, hesaplamalarda veya geri dönüşlerde kısa devre (short-circuit) işlemleri yapılmadığı doğrulanmalıdır; bu, bilgi sızıntısını önlemek için gereklidir.	3
11.2.5	Tüm kriptografik modüllerin güvenli şekilde hata verdiği ve hataların, padding oracle saldırıları gibi zafiyetleri mümkün kılmayacak şekilde ele alındığı doğrulanmalıdır.	3

V11.3 Şifreleme Algoritmaları

AES ve CHACHA20 üzerine kurulu kimliği doğrulanmış şifreleme algoritmaları, modern kriptografik uygulamaların temelini oluşturur.

#	Açıklama	Seviye
11.3.1	Güvensiz blok modlarının (örneğin, ECB) ve zayıf doldurma (padding) şemalarının (örneğin, PKCS#1 v1.5) kullanılmadığı doğrulanmalıdır.	1
11.3.2	Yalnızca AES ile GCM gibi onaylı şifreleme algoritmalarının ve modlarının kullanıldığı doğrulanmalıdır.	1
11.3.3	Şifrelenmiş verilerin yetkisiz değişikliklere karşı korunduğu doğrulanmalıdır. Tercihen bu koruma, onaylı bir kimliği doğrulanmış şifreleme yöntemi kullanılarak ya da onaylı bir şifreleme yöntemi ile onaylı bir MAC algoritmasının kombinasyonu ile sağlanmalıdır.	2

#	Açıklama	Seviye
11.3.4	Sayılar, başlatma vektörleri (IV) ve diğer tek kullanımlık değerlerin, aynı şifreleme anahtarı ve veri öğesi çifti için birden fazla kez kullanılmadığı doğrulanmalıdır. Oluşturma yöntemi kullanılan algoritmaya uygun olmalıdır.	3
11.3.5	Şifreleme algoritması ve MAC algoritması kombinasyonlarının "encrypt-then-MAC"modunda çalıştığı doğrulanmalıdır.	3

V11.4 Hashing ve Hash Tabanlı Fonksiyonlar

Kriptografik hash'ler, dijital imzalar, HMAC, anahtar türetme fonksiyonları (KDF), rastgele bit üretimi ve parola saklama gibi çok çeşitli kriptografik protokollerde kullanılır. Kriptografik sistemin güvenliği, kullanılan hash fonksiyonlarının güvenliği ile doğrudan ilişkilidir. Bu bölüm, kriptografik işlemlerde güvenli hash fonksiyonlarının kullanımına dair gereksinimleri özetler.

Parola saklama ve kriptografi ile ilgili ek için OWASP Password Storage Cheat Sheet belgesi de fayda ve rehberlik sağlar.

#	Açıklama	Seviye
11.4.1	Dijital imzalar, HMAC, KDF ve rastgele bit üretimi gibi genel kriptografik kullanım senaryolarında yalnızca onaylı hash fonksiyonlarının kullanıldığı doğrulanmalıdır. MD5 gibi yasaklı hash fonksiyonları hiçbir kriptografik amaçla kullanılmamalıdır.	1
11.4.2	Parolaların, onaylı, hesaplama açısından yoğun ve mevcut rehberliğe göre yapılandırılmış bir anahtar türetme fonksiyonu (diğer adıyla parola hash'leme fonksiyonu) kullanılarak saklandığı doğrulanmalıdır. Ayarlar, güvenlik ve performans dengesini sağlayarak kaba kuvvet saldırılarını zorlaştırmalıdır.	2
11.4.3	Dijital imzalarda kullanılan hash fonksiyonlarının, veri bütünlüğü ve kimlik doğrulama amacıyla kullanıldığında çakışmaya karşı dayanıklı ve yeterli bit uzunluğuna sahip olduğu doğrulanmalıdır. Çakışma dayanıklılığı gerekiyorsa çıktı uzunluğu en az 256 bit olmalıdır. Sadece ikinci ön-görüntü (2nd pre-iamge) saldırılarına karşı dayanıklılık gerekiyorsa çıktı uzunluğu en az 128 bit olmalıdır.	2

#	Açıklama	Seviye
11.4.4	Parolalardan gizli anahtarlar türetilirken, onaylı anahtar türetme fonksiyonlarının anahtar uzatma (key stretching) parametreleriyle birlikte kullanıldığı doğrulanmalıdır. Bu parametreler, kaba kuvvet saldırılarını önleyecek şekilde güvenlik ve performans dengesine sahip olmalıdır.	2

V11.5 Rastgele Değerler

Kriptografik olarak güvenli sözde rastgele sayı üretimini (Cryptographically secure Pseudo-random Number Generation - CSPRNG), doğru şekilde uygulamak son derece zordur. Genellikle, bir sistemdeki iyi entropi kaynakları aşırı kullanıldığında hızla tükenir. Düşük rastgelelik seviyesine sahip kaynaklar ise tahmin edilebilir anahtarlara ve sırların ortaya çıkmasına yol açabilir.

#	Açıklama	Seviye
11.5.1	Tahmin edilemez olması amaçlanan tüm rastgele sayıların ve dizelerin, CSPRNG ile oluşturulduğu ve en az 128 bit entropiye sahip olduğu doğrulanmalıdır. UUID'lerin bu koşulu dikkate almadığını unutmayın.	2
11.5.2	Kullanılan rastgele sayı üretim mekanizmasının, yoğun talep altında dahi güvenli çalışacak şekilde tasarlandığı doğrulanmalıdır.	3

V11.6 Açık Anahtar (Public Key) Kriptografisi

Birden fazla taraf arasında ortak bir sır paylaşımının mümkün olmadığı ya da arzu edilmediği durumlarda açık anahtar kriptografisi kullanılır.

Bu bağlamda, modern tehditlere karşı sistemin güvenliğini sağlamak için Diffie-Hellman ve Eliptik Eğri Diffie-Hellman (ECDH) gibi onaylı anahtar değişim mekanizmalarına ihtiyaç vardır. "Güvenli İletişim"bölümü, TLS için gereksinimleri sağlar; bu bölümdeki gereksinimler ise TLS dışındaki açık anahtar kriptografisi kullanım durumlarına yöneliktir.

#	Açıklama	Seviye
11.6.1	Anahtar üretimi, tohumlama (seeding) ve dijital imza üretimi ve doğrulama için yalnızca onaylı kriptografik algoritmalar ve çalışma modlarının kullanıldığı doğrulanmalıdır. Anahtar üretim algoritmaları, bilinen saldırılara karşı savunmasız anahtarlar üretmemelidir. Örneğin, Fermat çarpanlama yöntemine karşı savunmasız RSA anahtarları oluşturulmamalıdır.	2

#	Açıklama	Seviye
11.6.2	Anahtar değişimi için yalnızca onaylı kriptografik algoritmaların kullanıldığı doğrulanmalıdır (örneğin Diffie-Hellman). Kullanılan parametrelerin güvenli olması gerektiğine dikkat edilmelidir. Bu, anahtar kurulum sürecine yönelik saldırıları önleyecek ve araya girme (adversary-in-the-middle) ya da kriptografik kırılmaları engelleyecektir.	3

V11.7 Kullanım Halindeki Verilerin Kriptografisi

Veri işlenirken korunması çok önemlidir. Tam bellek şifreleme, veri iletiminde şifreleme ve verilerin kullanımdan hemen sonra mümkün olan en kısa sürede şifrelenmesi gibi teknikler önerilir.

#	Açıklama	Seviye
11.7.1	Hassas verilerin kullanım sırasında korunmasını sağlamak amacıyla, yetkisiz kullanıcıların veya süreçlerin bu verilere erişmesini engelleyen tam bellek şifrelemesinin kullanıldığı doğrulanmalıdır.	3
11.7.2	Veri asgariyetinin sağlandığı, yani işleme sırasında mümkün olan en az miktarda verinin ortaya çıktığı ve verilerin kullanımdan hemen sonra veya mümkün olan en kısa sürede şifrelendiği doğrulanmalıdır.	3

Referanslar

Daha fazla bilgi için:

- OWASP Web Security Testing Guide: Testing for Weak Cryptography
- OWASP Cryptographic Storage Cheat Sheet
- FIPS 140-3
- NIST SP 800-57

V12 Güvenli İletişim

Kontrol Amacı

Bu bölüm, hem bir son kullanıcı istemcisi ile backend hizmeti arasında hem de iç sistemler ile backend hizmetleri arasındaki veri iletimini korumak için uygulanması gereken özel mekanizmalara ilişkin gereksinimleri içerir.

Bu bölümde öne çıkan genel kavramlar şunlardır:

- İletişimlerin dış sistemlerle şifreli, ideal olarak iç sistemlerde de şifreli olması.
- Şifreleme mekanizmalarının, tercih edilen algoritmalar ve şifreler dahil olmak üzere en güncel rehberliklere göre yapılandırılması.
- Yetkisiz taraflarca iletişimin engellenmediğinden emin olmak amacıyla imzalı sertifikaların kullanılması.

Genel ilkelere ve en iyi uygulamalara ek olarak, ASVS ayrıca Ek C - Kriptografi Standartları bölümünde kriptografik güç hakkında daha derin teknik bilgiler sunar.

V12.1 Genel TLS Güvenlik Rehberi

Bu bölüm, TLS iletişimini güvenli hale getirmek için başlangıç seviyesinde rehberlik sağlar. TLS yapılandırmasının güncel araçlarla düzenli olarak gözden geçirilmesi gerekir.

Wildcard TLS sertifikalarının kullanımı doğası gereği güvensiz değildir, ancak aynı sertifikanın üretim, hazırlık (staging), geliştirme ve test gibi tüm ortamlarda kullanılması halinde sertifika güvenliğinin tehlikeye girmesi, bu ortamlarda çalışan uygulamaların da güvenlik postürünü olumsuz etkileyebilir. Mümkünse her ortam için ayrı TLS sertifikalarının kullanılması, bu sertifikaların güvenli bir şekilde yönetilmesi ve korunması gereklidir.

#	Açıklama	Seviye
12.1.1	Yalnızca TLS protokolünün en güncel ve önerilen sürümlerinin (örneğin TLS 1.2 ve TLS 1.3) etkinleştirildiği ve en son sürümün varsayılan olarak tercih edildiği doğrulanmalıdır.	1
12.1.2	Yalnızca önerilen şifre takımlarının (cipher suite) etkinleştirildiği ve en güçlü şifre takımlarının tercih edildiği doğrulanmalıdır. Seviye 3 (L3) uygulamalar yalnızca ileri gizlilik (forward secrecy) sağlayan şifre takımlarını desteklemelidir.	2
12.1.3	Uygulamanın, mTLS istemci sertifikalarını kimlik doğrulama veya yetkilendirme amacıyla kullanmadan önce bu sertifikaların güvendiği sertifikalar arasında olduğunu doğruladığı kontrol edilmelidir.	2
12.1.4	Online Certificate Status Protocol (OCSP) Stapling gibi geçerli bir sertifika iptal kontrolü yönteminin etkinleştirildiği ve yapılandırıldığı doğrulanmalıdır.	3
12.1.5	TLS el sıkışma (handshake) sürecinde Server Name Indication (SNI) gibi hassas meta verilerin ifşa edilmesini önlemek için, uygulamanın TLS yapılandırmasında Encrypted Client Hello (ECH) özelliğinin etkinleştirildiği doğrulanmalıdır.	3

V12.2 Dışa Açık Hizmetlerle HTTPS İletişimi

Uygulamanın dışa açtığı tüm HTTP trafiğinin şifreli olarak iletildiğinden ve bu iletişimlerde herkese açık olarak güvenilen sertifikaların kullanıldığından emin olunmalıdır.

#	Açıklama	Seviye
12.2.1	İstemci ile dışa açık, HTTP tabanlı hizmetler arasındaki tüm bağlantılarda TLS kullanıldığı ve bu bağlantıların güvensiz ya da şifrelenmemiş iletişime geri dönmediği doğrulanmalıdır.	1
12.2.2	Dışa açık hizmetlerin herkese açık olarak güvenilen TLS sertifikaları kullandığı doğrulanmalıdır.	1

V12.3 Genel Servisler Arası İletişim Güvenliği

Sunucu iletişimleri (hem iç hem dış) sadece HTTP ile sınırlı değildir. Diğer sistemlerle olan bağlantıların da güvenli olması gerekir ve ideal olarak TLS kullanılmalıdır.

#	Açıklama	Seviye
12.3.1	Uygulamaya gelen ve uygulamadan çıkan tüm bağlantılarda izleme sistemleri, yönetim araçları, uzaktan erişim ve SSH, ara yazılımlar, veritabanları, ana sistemler (mainframe), iş ortağı sistemleri ya da dış API'ler dahil olmak üzere TLS gibi şifreli bir protokolün kullanıldığı ve güvensiz ya da şifrelenmemiş protokollere geri dönülmediği doğrulanmalıdır.	2
12.3.2	TLS istemcilerinin bir TLS sunucusu ile iletişime geçmeden önce alınan sertifikaları doğruladığı kontrol edilmelidir.	2
12.3.3	Uygulama içindeki HTTP tabanlı iç hizmetler arasında gerçekleşen tüm bağlantılarda TLS ya da uygun başka bir aktarım şifreleme mekanizmasının kullanıldığı ve güvensiz ya da şifrelenmemiş iletişime geri dönülmediği doğrulanmalıdır.	2
12.3.4	İç hizmetler arasındaki TLS bağlantılarında güvenilir sertifikaların kullanıldığı doğrulanmalıdır. İç sistemlerde oluşturulmuş ya da kendinden imzalı (self-signed) sertifikalar kullanılıyorsa, bu sertifikalara güvenen hizmetlerin yalnızca belirli dahili sertifika otoritelerine (CA) ve belirli kendinden imzalı sertifikalara güvenecek şekilde yapılandırıldıkları doğrulanmalıdır.	2

#	Açıklama	Seviye
12.3.5	Sistem içindeki hizmetlerin birbiriyle iletişim kurarken her bir uç noktanın kimliğini doğrulamak için güçlü kimlik doğrulama yöntemlerinin kullanıldığı doğrulanmalıdır. Bu yöntemler arasında, açık anahtar altyapısı (public-key infrastructure) kullanan ve tekrar oynatma saldırılarına karşı dayanıklı olan TLS istemci kimlik doğrulaması gibi yöntemler yer almalıdır. Mikroservis mimarilerinde, sertifika yönetimini basitleştirmek ve güvenliği artırmak amacıyla bir servis ağı (service mesh) kullanılması değerlendirilmelidir.	3

Referanslar

Daha fazla bilgi için:

- OWASP - Transport Layer Security Cheat Sheet
- Mozilla's Server Side TLS configuration guide
- Mozilla's tool to generate known good TLS configurations.
- O-Saft - OWASP Project to validate TLS configuration

V13 Konfigürasyon

Kontrol Amacı

Uygulamanın varsayılan konfigürasyonu, İnternet üzerinde güvenli kullanım için uygun olmalıdır.

Bu bölüm, geliştirme, derleme ve dağıtım aşamalarında uygulanması gereken çeşitli konfigürasyonlara dair rehberlik sağlar.

Ele alınan konular veri sızıntısını önlemek, bileşenler arası iletişimi güvenli şekilde yönetmek ve gizli verileri korumayı içerir.

V13.1 Konfigürasyon Dokümantasyonu

Bu bölüm, uygulamanın iç ve dış hizmetlerle nasıl iletişim kurduğunu açıklayan dokümantasyon gereksinimlerini ve hizmetlerin erişilemez olması durumunda erişilebilirliğin kaybını önlemek için kullanılabilecek teknikleri tanımlar. Ayrıca, gizli verilerle ilgili belgeleri de kapsar.

#	Açıklama	Seviye
13.1.1	Uygulamanın tüm iletişim ihtiyaçlarının dokümante edildiği doğrulanmalıdır. Buna, uygulamanın bağımlı olduğu dış hizmetler ve son kullanıcının uygulamanın bağlantı kurmasını sağlayabileceği dış konumlar da dahil olmalıdır.	2
13.1.2	Uygulamanın kullandığı her bir hizmet için, dokümantasyonda eşzamanlı bağlantı sayısı (örn. bağlantı havuzu limitleri) ve bu sınır aşıldığında uygulamanın davranışı, geri dönüş veya kurtarma mekanizmaları da dahil olmak üzere hizmet reddi koşullarını önlemek için tanımlandığı doğrulanmalıdır.	3
13.1.3	Uygulamanın, kullandığı her bir dış sistem veya hizmet (örn. veritabanları, dosya tanıtıcıları, iş parçacıkları, HTTP bağlantıları) için kaynak yönetimi stratejilerini tanımladığı doğrulanmalıdır. Bu stratejiler, kaynak serbest bırakma prosedürlerini, zaman aşımı ayarlarını, hata durumlarını ve tekrar deneme mantığı uygulanmışsa sınırları, gecikmeleri ve geri çekilme (back-off) algoritmalarını içermelidir. Senkron HTTP istek–yanıt işlemleri için kısa zaman aşımı süreleri zorunlu tutulmalı ve tekrar eden denemeler devre dışı bırakılmalı ya da sıkı şekilde sınırlandırılmalıdır.	3
13.1.4	Uygulamanın, güvenlik açısından kritik olan gizli verileri tanımladığı ve kuruluşun tehdit modeli ve iş ihtiyaçlarına göre bu verilerin döndürülme (rotation) zamanlamasının belgelendiği doğrulanmalıdır.	3

V13.2 Backend İletişim Konfigürasyonu

Uygulamalar, API'ler, veritabanları veya diğer bileşenler gibi çok sayıda hizmetle etkileşime girer. Bu hizmetler, uygulamaya dâhil olarak kabul edilebilir ancak uygulamanın standart erişim kontrol mekanizmalarına dahil olmayabilir veya tamamen dış sistemler olabilir. Her iki durumda da, uygulamanın bu bileşenlerle güvenli şekilde iletişim kuracak şekilde yapılandırılması ve gerekiyorsa bu yapılandırmanın korunması gereklidir.

Not: "Güvenli İletişim"bölümü, aktarım sırasında şifreleme için rehberlik sağlar.

#	Açıklama	Seviye
13.2.1	Uygulamanın, standart kullanıcı oturum mekanizmasını desteklemeyen backend bileşenleri (örneğin API'ler, ara katmanlar, veri katmanları) arasında gerçekleşen iletişimlerin kimlik doğrulamasıyla korunduğu doğrulanmalıdır. Kimlik doğrulama; bireysel hizmet hesapları, kısa ömürlü token'lar ya da sertifika tabanlı kimlik doğrulama ile yapılmalıdır ve parola, API anahtarı ya da ayrıcalıklı erişime sahip paylaşımlı hesaplar gibi sabit kimlik bilgileri kullanılmamalıdır.	2
13.2.2	Uygulamanın backend bileşenleri arasında (yerel hizmetler, işletim sistemi hizmetleri, API'ler, ara katmanlar ve veri katmanları dahil) gerçekleştirilen iletişimlerin, yalnızca gerekli en az ayrıcalığa sahip hesaplarla yapıldığı doğrulanmalıdır.	2
13.2.3	Bir hizmet kimlik doğrulaması için bir kimlik bilgisi kullanılması gerekiyorsa, bu kimlik bilgisinin varsayılan bir değer (örneğin root/root veya admin/admin) olmadığının doğrulanması gereklidir.	2
13.2.4	Uygulamanın iletişim kurmasına izin verilen dış kaynaklar ya da sistemlerin, bir allowlist (izinli liste) ile tanımlandığı doğrulanmalıdır. Bu liste, uygulama katmanında, web sunucusunda, güvenlik duvarında veya farklı katmanların birleşiminde uygulanabilir.	2
13.2.5	Web veya uygulama sunucusunun, dış sistemlere veri veya dosya gönderimi veya yüklemesi yaparken sadece belirli kaynaklara izin verecek şekilde allowlist ile yapılandırıldığı doğrulanmalıdır.	2
13.2.6	Uygulamanın farklı hizmetlere bağlantı kurduğu durumlarda, bu bağlantılar için belgelenmiş yapılandırmaları (örn. maksimum eşzamanlı bağlantı, bağlantı limiti aşıldığında davranış, zaman aşımı, tekrar deneme stratejileri) takip ettiğinin doğrulanması gereklidir.	3

V13.3 Gizli Bilgi Yönetimi

Gizli bilgilerin (secret) yönetimi, uygulama tarafından kullanılan verilerin korunması açısından temel bir konfigürasyon görevidir. Kriptografi ile ilgili özel gereksinimler "Kriptografi"bölümünde yer almakla birlikte bu bölüm, gizli verilerin yönetimi ve işlenmesine odaklanır.

#	Açıklama	Seviye
13.3.1	Parolalar, anahtar materyalleri, veritabanı ve üçüncü parti sistemlerle entegrasyon bilgileri, zaman tabanlı token'lar için tohum (seed) bilgileri, dahili gizli veriler ve API anahtarları gibi backend sırlarının güvenli bir şekilde oluşturulması, saklanması, erişim denetiminin sağlanması ve imha edilmesi için bir gizli bilgi yönetim çözümünün (örneğin key vault) kullanıldığı doğrulanmalıdır. Bu gizli bilgiler uygulama kaynak koduna dahil edilmemeli ve yapı çıktılarında yer almamalıdır. Seviye 3 uygulamalar için donanım destekli bir çözüm (örneğin HSM) gereklidir.	2
13.3.2	Gizli bilgilere erişimin en az ayrıcalık prensibine uygun şekilde yapılandırıldığı doğrulanmalıdır.	2
13.3.3	Tüm kriptografik işlemlerin, anahtar materyallerinin güvenli bir şekilde yönetilmesi ve dışarıya sızmasının engellenmesi amacıyla, izole edilmiş bir güvenlik modülü (örneğin vault ya da donanım güvenlik modülü) aracılığıyla gerçekleştirildiği doğrulanmalıdır.	3
13.3.4	Uygulamanın belgelerinde belirtildiği şekilde gizli bilgilerin süresinin dolacak şekilde yapılandırıldığı ve düzenli olarak döndürüldüğü doğrulanmalıdır.	3

V13.4 İstenmeyen Bilgi Sızıntısı

Üretim yapılandırmaları, gereksiz veri ifşasını önleyecek şekilde sertleştirilmiş (hardened) olmalıdır. Bu tür güvenlik açıkları genellikle tek başına büyük risk olarak değerlendirilmese de diğer açıklıklarla zincirlenerek sömürülebilir. Varsayılan olarak bu açıklıkların bulunmaması, uygulamaya yapılacak saldırıların zorluk seviyesini artırır.

Örneğin, sunucu taraflı bileşenlerin sürüm bilgisini gizlemek, tüm bileşenlerin yamanması ihtiyacını ortadan kaldırmaz; dizin listelemeyi devre dışı bırakmak, yetkilendirme kontrolleri ihtiyacını ya da dosyaların herkese açık klasörden uzak tutulması gerekliliğini ortadan kaldırmaz. Ancak bunlar, saldırıların başarıya ulaşma olasılığını azaltır.

#	Açıklama	Seviye
13.4.1	Uygulamanın .git veya .svn klasörleri gibi kaynak kontrolüyle ilgili metadata içermeyecek şekilde deploy edildiğini ya da bu klasörlerin hem dış erişime hem de uygulama erişimine kapalı olduğu doğrulanmalıdır.	1
13.4.2	Tüm bileşenler için üretim ortamlarında hata ayıklama (debug) modlarının devre dışı bırakıldığı doğrulanmalıdır.	2

#	Açıklama	Seviye
13.4.3	Web sunucularının, açık şekilde istenmediği sürece dizin listelemesi (directory listing) özelliğini istemcilere sunmadığı doğrulanmalıdır.	2
13.4.4	HTTP TRACE metodunun üretim ortamlarında desteklenmediği, böylece potansiyel bilgi sızıntılarının önlendiği doğrulanmalıdır.	2
13.4.5	Belgeler (örneğin dahili API belgeleri) ve izleme uç noktalarının (monitoring endpoints) yalnızca açıkça belirtildiği durumlarda erişilebilir olduğu doğrulanmalıdır.	2
13.4.6	Uygulamanın arka uç bileşenlerinin ayrıntılı sürüm bilgilerini ifşa etmediği doğrulanmalıdır.	3
13.4.7	Web katmanının yalnızca belirli dosya uzantılarına sahip dosyaları sunacak şekilde yapılandırıldığı, bu sayede istem dışı bilgi, yapılandırma ya da kaynak kod sızıntılarının önlendiği doğrulanmalıdır.	3

Referanslar

Daha fazla bilgi için:

- OWASP Web Security Testing Guide: Configuration and Deployment Management Testing

V14 Veri Koruma

Kontrol Amacı

Uygulamalar, tüm kullanım senaryolarını ve kullanıcı davranışlarını önceden öngöremez. Bu nedenle, istemci cihazlarda hassas verilere yetkisiz erişimi sınırlamak için kontroller uygulanmalıdır.

Bu bölüm, hangi verilerin korunması gerektiğini tanımlama, nasıl korunacağını belirleme ve uygulanması gereken özel mekanizmalar ile kaçınılması gereken hatalarla ilgili gereksinimleri içerir.

Veri koruma ile ilgili bir diğer husus da toplu veri çıkarımı, değiştirme veya aşırı kullanımdır. Her sistemin gereksinimleri muhtemelen çok farklı olacaktır; bu nedenle, neyin “anormal”olduğunun belirlenmesi tehdit modeli ve iş riski dikkate alınarak yapılmalıdır. ASVS perspektifinden bakıldığında, bu tür sorunların tespiti “Güvenlik Günlüğü Tutma ve Hata Yönetimi”bölümünde, sınırlamaların belirlenmesi ise “Doğrulama ve İş Mantığı”bölümünde ele alınmaktadır.

V14.1 Veri Koruma Dokümantasyonu

Verilerin korunabilmesi için temel ön koşullardan biri, hangi verilerin hassas olarak kabul edilmesi gerektiğinin kategorize edilmesidir. Hassasiyetin birkaç farklı düzeyi olabilir ve her düzey için uygulanması gereken kontroller farklılık gösterebilir.

Verilerin saklanması, kullanımı ve iletimi konusunda uygulamaların uyması gereken çeşitli gizlilik düzenlemeleri ve yasalar bulunmaktadır. Bu bölüm artık bu tür yasal düzenlemeleri çoğaltmaya çalışmamakta, bunun yerine hassas verileri korumaya yönelik temel teknik hususlara odaklanmaktadır. Lütfen yerel yasa ve düzenlemelere başvurun ve gerektiğinde nitelikli bir gizlilik uzmanı veya avukata danışın.

#	Açıklama	Seviye
14.1.1	Uygulama tarafından oluşturulan ve işlenen tüm hassas verilerin tanımlanıp, koruma seviyelerine göre sınıflandırıldığından emin olun. Bu, yalnızca kodlanmış (örneğin Base64 stringleri ya da JWT içindeki düz metin yükler gibi) ve bu nedenle kolayca çözülebilir verileri de içermelidir. Koruma seviyeleri, uygulamanın uymakla yükümlü olduğu veri koruma ve gizlilik düzenlemeleri ile standartlarını dikkate almalıdır.	2
14.1.2	Tüm hassas veri koruma seviyeleri için belgelenmiş bir koruma gereksinimleri seti tanımlandığından emin olun. Bu gereksinimler genel şifreleme, bütünlük doğrulama, saklama süresi, verinin nasıl günlüğe kaydedileceği, günlüklerdeki hassas verilere erişim kontrolleri, veritabanı düzeyinde şifreleme, gizlilik ve gizliliği artırıcı teknolojiler ile diğer gizlilik gereksinimlerini kapsamalıdır.	2

V14.2 Genel Veri Koruması

Bu bölüm, veri korumaya yönelik çeşitli pratik gereksinimleri içerir. Bunların çoğu, istenmeyen veri sızıntısı gibi belirli konulara yöneliktir. Ancak aynı zamanda her bir veri öğesi için gerekli olan koruma düzeyine göre koruma kontrollerinin uygulanmasına ilişkin genel bir gereksinim de bulunmaktadır.

#	Açıklama	Seviye
14.2.1	Hassas verilerin yalnızca HTTP message body veya header alanlarında sunucuya gönderildiği ve URL ile sorgu parametrelerinin, API anahtarı veya session token'ı gibi hassas bilgiler içermediği doğrulanmalıdır.	1

#	Açıklama	Seviye
14.2.2	Hassas verilerin, yük dengeleyiciler ve uygulama önbellekleri gibi sunucu bileşenlerinde önbelleğe alınmasının engellendiği veya kullanım sonrası bu verilerin güvenli biçimde temizlendiği doğrulanmalıdır.	2
14.2.3	Uygulama kontrolü dışında veri toplanmasını engellemek için, tanımlanmış hassas verilerin, kullanıcı izleyicileri gibi güvensiz taraflara gönderilmediği doğrulanmalıdır.	2
14.2.4	Hassas verilere yönelik şifreleme, bütünlük doğrulama, veri saklama, loglama yöntemi, loglardaki erişim kontrolleri, gizlilik teknolojileri gibi kontrollerin, söz konusu verinin belgelenmiş koruma seviyesinde tanımlandığı şekilde uygulandığı doğrulanmalıdır.	2
14.2.5	Önbellekleme mekanizmalarının yalnızca belirli içerik türüne sahip yanıtları önbelleğe alacak şekilde yapılandırıldığı ve hassas, dinamik içeriklerin önbelleğe alınmadığı doğrulanmalıdır. Web sunucusu, mevcut olmayan bir dosya talebinde geçerli ama farklı bir dosya döndürmek yerine 404 veya 302 cevabı vermelidir. Bu, Web Cache Deception saldırılarını önlemeye yardımcı olur.	3
14.2.6	Uygulamanın yalnızca işlevsellik için gerekli olan asgari hassas veriyi döndürdüğü doğrulanmalıdır. Örneğin, kredi kartı numarasının yalnızca bazı rakamlarının döndürülmesi; tam numaranın döndürülmesi gerekiyorsa, kullanıcı arayüzünde gizlenmiş (maskelenmiş) şekilde sunulması gerekir, aksi kullanıcı özel olarak görüntülemek istemedikçe gösterilmemelidir.	3
14.2.7	Hassas bilgilerin, veri saklama sınıflandırmasına tabi tutulduğu, geçerliliğini yitirmiş veya artık gerekli olmayan verilerin belirlenmiş bir zaman çizelgesiyle ya da gerektiğinde otomatik olarak silindiği doğrulanmalıdır.	3
14.2.8	Kullanıcı tarafından yüklenen dosyaların meta verilerinden hassas bilgilerin çıkarıldığı veya bu bilgilerin yalnızca kullanıcıdan açık onay alındığında saklandığı doğrulanmalıdır.	3

V14.3 İstemci Tarafı Veri Koruması

Bu bölüm, uygulamanın istemci veya kullanıcı aracısı tarafında verilerin belirli yollarla sızmasını önlemeye yönelik gereksinimleri içerir.

#	Açıklama	Seviye
14.3.1	Kimliği doğrulanmış verilerin, istemci veya oturum sonlandığında istemci depolamasından (örneğin tarayıcı DOM'undan) temizlendiği doğrulanmalıdır. 'Clear-Site-Data'HTTP response header bu konuda yardımcı olabilir ancak istemci tarafı, sunucu bağlantısı mevcut olmadığında da temizlik işlemini yapabiliyor olmalıdır.	1
14.3.2	Uygulamanın, tarayıcılarda hassas verilerin önbelleğe alınmasını önlemek amacıyla yeterli anti-önbellekleme HTTP response header (örneğin Cache-Control: no-store) ayarladığı doğrulanmalıdır.	2
14.3.3	Tarayıcıda depolanan verilerin (örneğin localStorage, sessionStorage, IndexedDB veya çerezler) hassas veri içermediği, yalnızca session token' larının istisna olarak kabul edildiği doğrulanmalıdır.	2

Referanslar

Daha fazla bilgi için:

- Güvenlik ve önbelleğe alma karşıtı header field'ları kontrol etmek için Security Headers web sitesini kullanmayı değerlendirebilirsiniz
- Mozilla'nın önbelleğe alma karşıtı header'larla ilgili dokümantasyonu
- OWASP Secure Headers project
- OWASP Privacy Risks Project
- OWASP User Privacy Protection Cheat Sheet
- Australian Privacy Principle 11 - Security of personal information
- European Union General Data Protection Regulation (GDPR) overview
- European Union Data Protection Supervisor - Internet Privacy Engineering Network
- "Clear-Site-Data"header'ıyla ilgili bilgi
- Web Cache Deception hakkında white paper

V15 Güvenli Kodlama ve Mimari

Kontrol Amacı

ASVS gereksinimlerinin birçoğu kimlik doğrulama veya yetkilendirme gibi belirli bir güvenlik alanına ya da loglama veya dosya işleme gibi belirli uygulama işlevlerine ilişkindir.

Bu bölüm, uygulamalar tasarlanırken ve geliştirilirken dikkate alınması gereken genel güvenlik gereksinimlerini kapsar. Bu gereksinimler yalnızca temiz mimari ve kod kalitesine değil, aynı

zamanda uygulama güvenliği için gerekli olan belirli mimari ve kodlama uygulamalarına da odaklanır.

V15.1 Güvenli Kodlama ve Mimari Dokümantasyonu

Güvenli ve savunulabilir bir mimarinin oluşturulmasına yönelik birçok gereksinim, uygulamada kullanılan bileşenler ve belirli güvenlik kontrollerinin uygulanmasına ilişkin kararların açıkça belgelenmesine dayanır.

Bu bölüm, "tehlikeli işlevsellik"içeren bileşenler veya "riskli bileşenler"olarak kabul edilen parçaların tanımlanması dahil olmak üzere, dokümantasyon gereksinimlerini açıklar.

"Tehlikeli işlevsellik"içeren bir bileşen, güvenilmeyen verilerin serisizleştirmesini (deserialization) yapan, ham dosya veya ikili veri ayrıştırması gerçekleştiren, dinamik kod çalıştıran veya doğrudan bellekle etkileşen bir dahili ya da üçüncü taraf bileşen olabilir. Bu tür işlemlerdeki güvenlik açıkları, uygulamanın ve altında yatan altyapının ele geçirilmesine yol açabilecek yüksek risklidir.

"Riskli bileşen", güvenlik kontrolleri eksik ya da zayıf şekilde uygulanmış bir üçüncü taraf (ekip içinde geliştirilmemiş) kütüphanedir. Örnekler arasında kötü şekilde sürdürülen, desteklenmeyen, ömrünün sonuna gelmiş veya ciddi güvenlik açıkları geçmişi olan bileşenler yer alır.

Bu bölüm aynı zamanda üçüncü taraf bileşenlerdeki güvenlik açıklarının giderilmesi için uygun zaman çerçevelerinin tanımlanmasının önemini vurgular.

#	Açıklama	Seviye
15.1.1	Güvenlik açığı içeren üçüncü taraf bileşen sürümleri ve genel olarak kütüphane güncellemeleri için, uygulama dokümantasyonunda risk bazlı düzeltme zamanlarının tanımlandığı doğrulanmalıdır.	1
15.1.2	Yazılım bileşen listesi (Software Bill of Materials - SBOM) gibi bir envanterin tutulduğu, kullanılan tüm üçüncü taraf kütüphanelerin önceden tanımlanmış, güvenilir ve düzenli olarak güncellenen depolardan alındığı doğrulanmalıdır.	2
15.1.3	Uygulama dokümantasyonunda zaman veya kaynak açısından yoğun işlevlerin tanımlandığı, bu işlevlerin aşırı kullanımı durumunda kullanılabilirlik kaybının nasıl önleneceği ve yanıt oluşturma süresinin istemci zaman aşımını aşmaması için hangi önlemlerin alındığı belirtilmelidir. Örnek savunmalar kuyruklama, eşzamansız işleme, kullanıcı başına ve uygulama genelinde paralel işlem sınırlarını içerebilir.	2
15.1.4	Uygulama dokümantasyonunda "riskli bileşenler"olarak değerlendirilen üçüncü taraf kütüphanelerin vurgulandığı doğrulanmalıdır.	3

#	Açıklama	Seviye
15.1.5	Uygulama dokümantasyonunda “tehlikeli işlevsellik”içeren uygulama bölümlerinin açıkça belirtildiği doğrulanmalıdır.	3

V15.2 Güvenlik Mimarisi ve Bağımlılıklar

Bu bölüm riskli, güncel olmayan veya güvensiz bağımlılıkların ve bileşenlerin bağımlılık yönetimi yoluyla nasıl ele alınması gerektiğine dair gereksinimleri içerir.

Ayrıca, “tehlikeli işlemler”veya “riskli bileşenlerin”(önceki bölümde tanımlandığı şekilde) kullanımının etkisini azaltmak ve kaynak yoğun işlevlerin aşırı kullanımından kaynaklı kullanılabilirlik kayıplarını önlemek amacıyla sandboxing, kapsülleme (encapsulation), konteynerleştirme ve ağ izolasyonu gibi mimari düzeydeki tekniklerin kullanımını da kapsar.

#	Açıklama	Seviye
15.2.1	Uygulamanın yalnızca, belgelenmiş güncelleme ve düzeltme sürelerini aşmamış bileşenleri içerdiği doğrulanmalıdır.	1
15.2.2	Uygulamanın, belgelenmiş güvenlik kararları ve stratejilere dayanarak, zaman veya kaynak açısından yoğun işlevlerin neden olabileceği kullanılabilirlik kaybına karşı savunmalar uyguladığı doğrulanmalıdır.	2
15.2.3	Üretim ortamının yalnızca uygulamanın çalışması için gerekli işlevleri içerdiği ve test kodları, örnek kod parçaları veya geliştirme işlevleri gibi gereksiz işlevsellikleri açığa çıkarmadığı doğrulanmalıdır.	2
15.2.4	Üçüncü taraf bileşenlerin ve bunların tüm transitif bağımlılıklarının beklenen depolardan (iç kaynaklı ya da harici) alındığı ve bağımlılık karışıklığı (dependency confusion) saldırılarına açık olmadığı doğrulanmalıdır.	3
15.2.5	Uygulamanın, “tehlikeli işlevsellik”içeren veya “riskli bileşenler”kullanan bölümlerine ek korumalar uyguladığı doğrulanmalıdır. Bu korumalar sandbox, kapsülleme, konteynerleştirme veya ağ düzeyinde izolasyon gibi teknikler aracılığıyla, bir uygulama bölümünün ele geçirilmesi durumunda saldırganın diğer bölümlere sıçramasını geciktirmeyi veya engellemeyi amaçlamalıdır.	3

V15.3 Savunmacı Kodlama

Bu bölüm, belirli bir programlama dilinde güvensiz kodlama kalıplarının kullanılması sonucu oluşan tür oynama (type juggling), prototip zehirleme (prototype pollution) gibi güvenlik açıklarını kapsar. Bazıları tüm dillere uygun olmayabilirken, diğerleri dile özel düzeltmelere sahip olabilir ya da HTTP parametrelerinin işlenmesi gibi framework'e özgü davranışlarla ilişkilidir. Ayrıca uygulama güncellemelerinin kriptografik olarak doğrulanmamasıyla ilgili riskleri de ele alır.

Ayrıca, veri öğelerini temsil etmek için nesnelerin kullanılması ve bu nesnelerin dış API'ler üzerinden alınması ve döndürülmesi durumunda ortaya çıkan riskleri de dikkate alır. Bu durumda, uygulama kullanıcı girdileriyle değiştirilmemesi gereken alanların değiştirilememesini (kitlesel atama –mass assignment) sağlamalı ve hangi alanların döndürüldüğü konusunda seçici olmalıdır. Alan erişimi bir kullanıcının izinlerine bağlıysa, bu durum "Authorization"bölümündeki domain düzeyinde erişim kontrolü gereksinimi bağlamında değerlendirilmelidir.

#	Açıklama	Seviye
15.3.1	Uygulamanın yalnızca gerekli alanların bir alt kümesini veri nesnesinden döndürdüğü doğrulanmalıdır. Örneğin, bazı alanlara kullanıcıların erişmemesi gerektiğinden tüm veri nesnesi döndürülmemelidir.	1
15.3.2	Uygulama backend'i harici URL'lere çağrı yaptığında, yalnızca amaçlanan işlevsellik söz konusuysa yönlendirmeleri takip edecek şekilde yapılandırıldığının doğrulanması gerekir.	2
15.3.3	Uygulamanın, kitlesel atama saldırılarına karşı, her bir controller ve eylem için izin verilen alanları sınırlandırarak savunmalar sağladığı doğrulanmalıdır. Örneğin, bir alan değeri, bu işlemde değiştirilmesi amaçlanmadıysa eklenememeli veya güncellenememelidir.	2
15.3.4	Tüm proxy ve ara katman bileşenlerinin kullanıcının orijinal IP adresini, uç kullanıcı tarafından değiştirilemeyecek güvenilir veri alanları kullanarak doğru şekilde ilettiği ve uygulamanın ve web sunucusunun bu doğru değeri, loglama ve hız sınırlama gibi güvenlik kararları için kullandığı doğrulanmalıdır. Ayrıca, orijinal IP adresinin bile dinamik IP'ler, VPN'ler veya kurumsal güvenlik duvarları nedeniyle güvenilir olmayabileceği dikkate alınmalıdır.	2
15.3.5	Uygulamanın değişkenlerin doğru türde olduğunu açıkça doğruladığı ve sıkı eşitlik ve karşılaştırma işlemleri kullandığı doğrulanmalıdır. Bu, değişken türüyle ilgili varsayımlardan kaynaklanan tür oynama (type juggling) veya tür karışıklığı (type confusion) güvenlik açıklarını önlemek içindir.	2
15.3.6	JavaScript kodunun prototip zehirleme (prototype pollution) saldırılarını önleyecek şekilde yazıldığı, örneğin, nesne literal'leri yerine Set() veya Map() gibi yapılar kullanıldığı doğrulanmalıdır.	2

#	Açıklama	Seviye
15.3.7	Uygulamanın, özellikle framework'ün istek parametrelerinin kaynağı (sorgu dizgisi, gövde parametreleri, çerezler veya başlık alanları) arasında ayrım yapmadığı durumlarda, HTTP parametre kirliliği (HTTP parameter pollution) saldırılarına karşı savunmalar uyguladığı doğrulanmalıdır.	2

V15.4 Güvenli Eş Zamanlılık (Concurrency)

Yarış durumları (race condition), kontrol anından kullanım anına (TOCTOU - time-of-check to time-of-use) açıkları, deadlock, livelock, iş parçacığı (thread) açlığı ve hatalı senkronizasyon gibi eş zamanlılık sorunları, öngörülemeyen davranışlara ve güvenlik risklerine yol açabilir. Bu bölüm, bu riskleri azaltmaya yardımcı olacak çeşitli teknikleri ve stratejileri içerir.

#	Açıklama	Seviye
15.4.1	Çok iş parçacıklı kodda (örneğin cache'ler, dosyalar veya birden fazla iş parçacığı tarafından erişilen bellek içi nesneler gibi) paylaşılan nesnelere yalnızca thread-safe (iş parçacığı güvenli) türler ve kilitleme (lock) veya semaphore gibi senkronizasyon mekanizmaları kullanılarak erişildiği doğrulanmalıdır. Bu, yarış durumlarını ve veri bozulmasını önlemeye yöneliktir.	3
15.4.2	Bir kaynağın durumu üzerindeki kontroller (örneğin varlığı ya da izinleri gibi) ile bu kontrole bağlı eylemlerin tek bir atomik işlem olarak gerçekleştirildiği doğrulanmalıdır. Bu, TOCTOU (time-of-check to time-of-use) yarış durumlarını önlemeye yöneliktir. Bir dosyanın varlığının kontrol edilip ardından açılması veya bir kullanıcının erişiminin doğrulanıp ardından yetki verilmesi gibi işlemler buna örnektir.	3
15.4.3	Kilitlerin (locks) tutarlı biçimde kullanıldığı, iş parçacıklarının birbirini beklemesi ya da sonsuz döngüye girmesi (deadlock veya livelock) gibi durumların önlendiği doğrulanmalıdır. Ayrıca, kilitleme mantığı, kaynağı yöneten kod içinde kalmalı ve dış sınıflar veya dış kodlar tarafından istemeden veya kötü niyetli şekilde değiştirilememelidir.	3
15.4.4	Kaynak tahsis politikalarının, iş parçacığı açlığını önleyecek şekilde, kaynaklara adil erişimi sağladığı doğrulanmalıdır. Örneğin, thread pool (iş parçacığı havuzu) kullanımı, düşük öncelikli iş parçacıklarının da makul sürelerde çalışabilmesi için uygulanmalıdır.	3

Referanslar

Daha fazla bilgi için:

- OWASP Prototype Pollution Prevention Cheat Sheet
- OWASP Mass Assignment Prevention Cheat Sheet
- OWASP CycloneDX Bill of Materials Specification
- OWASP Web Security Testing Guide: Testing for HTTP Parameter Pollution

V16 Güvenlik Günlüklemesi (Logging) ve Hata Yönetimi

Kontrol Amacı

Güvenlik logları hata ya da performans loglarından farklı olarak kimlik doğrulama kararları, erişim kontrol kararları ve giriş doğrulama ya da iş mantığı doğrulama gibi güvenlik kontrollerini aşma girişimleri gibi güvenlikle ilgili olayları kaydetmek için kullanılır. Bu logların amacı tespit, müdahale ve inceleme süreçlerini desteklemek için SIEM gibi analiz araçlarına uygun, yüksek sinyalli ve yapılandırılmış veri sağlamaktır.

Loglar, yasal bir zorunluluk olmadıkça hassas kişisel verileri içermemeli ve kaydedilen tüm veriler yüksek değerli birer varlık olarak korunmalıdır. Loglama işlemleri gizliliği veya sistem güvenliğini tehlikeye atmamalıdır. Uygulamalar ayrıca, hata durumlarında gereksiz bilgi sızdırma veya kesinti olmaksızın güvenli şekilde başarısız olmalıdır.

Detaylı uygulama rehberliği için referanslar bölümündeki OWASP Cheat Sheet dökümanlarına başvurulabilir.

V16.1 Güvenlik Loglaması Dokümantasyonu

Bu bölüm, uygulama katmanları genelinde gerçekleştirilen loglamanın açık ve eksiksiz bir envanterini oluşturmayı amaçlar. Bu, etkili güvenlik izleme, olay müdahalesi ve uyumluluk için gereklidir.

#	Açıklama	Seviye
16.1.1	Uygulamanın teknoloji yığını boyunca her katmanda hangi olayların loglandığını, log formatlarını, logların nereye yazıldığını, nasıl kullanıldığını, erişimin nasıl kontrol edildiğini ve logların ne kadar süre saklandığını belgeleyen bir envanterin mevcut olduğu doğrulanmalıdır.	2

V16.2 Genel Loglama

Bu bölüm, güvenlik loglarının tutarlı şekilde yapılandırılmasını ve beklenen meta verileri içermesini sağlamaya yönelik gereksinimleri kapsar. Amaç, logların makine tarafından okunabilir ve dağıtık sistemler ile araçlarda analiz edilebilir olmasını sağlamaktır.

Güvenlik olayları çoğunlukla hassas veriler içerir. Bu tür verilerin dikkatsizce kaydedilmesi durumunda, loglar da hassas veri niteliği kazanır ve bu da şifreleme zorunluluğu, daha sıkı saklama politikaları ve denetimlerde açıklanma riski gibi sonuçlara yol açar.

Bu nedenle yalnızca gerekli verilerin loglanması ve log verilerinin diğer hassas varlıklar gibi korunması kritik önemdedir.

Aşağıdaki gereksinimler metadata'nın loglanması, senkronizasyon, biçim ve kontrol için temel gereksinimleri belirler.

#	Açıklama	Seviye
16.2.1	Her bir log girdisinin, olayın ne zaman, nerede, kim tarafından ve ne olduğuna dair detaylı bir zaman çizelgesi oluşturulmasına olanak tanıyacak gerekli meta verileri içerdiği doğrulanmalıdır.	2
16.2.2	Tüm loglama bileşenlerinin zaman kaynaklarının senkronize olduğu ve güvenlik olayı meta verisindeki zaman damgalarının UTC kullanarak ya da açık zaman dilimi ofsetiyle belirtildiği doğrulanmalıdır. UTC kullanımı, özellikle yaz saati uygulamaları nedeniyle oluşabilecek karışıklıkları önlemek için tavsiye edilir.	2
16.2.3	Uygulamanın yalnızca log envanterinde belgelenmiş dosya ve hizmetlere log yazdığı ya da ilettiği doğrulanmalıdır.	2
16.2.4	Logların kullanılan log işleyici (log processor) tarafından okunabildiği ve ilişkilendirilebildiği doğrulanmalıdır. Tercihen yaygın bir log formatı kullanılmalıdır.	2
16.2.5	Hassas veriler loglanırken, uygulamanın veri koruma seviyesine uygun olarak loglama yaptığı doğrulanmalıdır. Örneğin kimlik bilgileri veya ödeme detayları gibi bazı verilerin loglanmasına izin verilmemelidir. Session token'ları gibi veriler ise yalnızca tamamen ya da kısmen maskelenmiş ya da hash'lenmiş olarak loglanmalıdır.	2

V16.3 Güvenlik Olayları

Bu bölüm, uygulama içerisinde güvenlikle ilgili olayların loglanmasına dair gereksinimleri tanımlar. Bu olayların yakalanması şüpheli davranışların tespiti, olay incelemelerinin desteklenmesi ve uyumluluk yükümlülüklerinin yerine getirilmesi açısından kritiktir.

Buradaki olay türleri örnek niteliğindedir. Her uygulama, kendine özgü risk faktörleri ve operasyonel bağlama sahiptir.

Not: ASVS, güvenlik olaylarının loglanmasını kapsam içine alır. Ancak SIEM kuralları veya izleme altyapısı gibi uyarı ve korelasyon sistemleri kapsam dışıdır.

#	Açıklama	Seviye
16.3.1	Tüm kimlik doğrulama işlemlerinin (başarılı ve başarısız girişimler dahil) loglandığı doğrulanmalıdır. Kullanılan kimlik doğrulama türü veya faktörleri gibi ek meta veriler de toplanmalıdır.	2
16.3.2	Başarısız yetkilendirme girişimlerinin loglandığı doğrulanmalıdır. Seviye 3 uygulamalar için, tüm yetkilendirme kararlarının ve hassas verilere erişim olaylarının (verinin kendisi olmadan) loglandığı da doğrulanmalıdır.	2
16.3.3	Uygulamanın, belgelenmiş güvenlik olaylarını ve güvenlik kontrollerini aşma girişimlerini (girdi doğrulama, iş mantığı, otomasyon karşıtı kontroller vb.) logladığı doğrulanmalıdır.	2
16.3.4	Beklenmeyen hatalar ve güvenlik kontrolü başarısızlıkları (örneğin backend TLS hataları gibi) loglandığı doğrulanmalıdır.	2

V16.4 Log Koruması

Loglar, olay incelemeleri ve adli analizler için değerli kanıtlardır ve korunmaları gerekir. Kolayca silinebilen ya da değiştirilebilen loglar güvenilirliğini kaybeder. Ayrıca loglar, uygulamanın iç davranışları veya hassas meta veriler hakkında bilgi barındırabileceğinden saldırganlar için cazip hedeflerdir.

Bu bölüm, logların yetkisiz erişimden, değiştirilmeden ve açıklanmaktan korunmasını ve güvenli, izole sistemlere aktarılmasını sağlamaya yönelik gereksinimleri tanımlar.

#	Description	Level
16.4.1	Tüm loglama bileşenlerinin, log enjeksiyonunu (log injection) önlemek amacıyla verileri uygun biçimde encode ettiği doğrulanmalıdır.	2
16.4.2	Logların yetkisiz erişime karşı korunduğu ve değiştirilemez olduğu doğrulanmalıdır.	2
16.4.3	Logların analiz, tespit, uyarı ve yükseltme amacıyla mantıksal olarak ayrı bir sisteme güvenli biçimde iletildiği doğrulanmalıdır. Böylece uygulama ihlal edilse bile logların tehlikeye girmesi önlenmiş olur.	2

V16.5 Hata Yönetimi

Bu bölüm, uygulamaların beklenmedik durumlarda güvenli şekilde başarısız olmasını ve hassas içsel detayları açıklamamasını sağlamak için gerekli gereksinimleri tanımlar.

#	Açıklama	Seviye
16.5.1	Beklenmedik ya da güvenlikle ilgili bir hata oluştuğunda, tüketiciye genel bir mesaj döndürüldüğü ve yığın izleri (stack trace), sorgular, gizli anahtarlar, token'lar gibi hassas sistem içi verilerin açıklanmadığı doğrulanmalıdır.	2
16.5.2	Uygulamanın, dış kaynaklara erişim başarısız olduğunda bile güvenli şekilde çalışmaya devam ettiği doğrulanmalıdır. Örneğin, circuit breaker ya da graceful degradation gibi desenler kullanılmalıdır.	2
16.5.3	Uygulamanın hata durumlarında güvenli ve düzenli şekilde başarısız olduğu doğrulanmalıdır. Bu, örneğin doğrulama hatalarına rağmen işlemlerin işlenmesine yol açabilecek fail-open durumlarını önlemelidir.	2
16.5.4	Ele alınmamış tüm istisnaları yakalayacak şekilde tanımlanmış bir "son çare hata yöneticisi"bulunduğu doğrulanmalıdır. Bu, hem loglara gitmesi gereken hata detaylarının kaybolmaması hem de tüm uygulama sürecinin çökerek hizmet dışı kalmasının önlenmesi açısından önemlidir.	3

Not: Swift, Go ve birçok fonksiyonel dilde olduğu gibi bazı diller, istisnaları veya "son çare"hata yöneticilerini desteklemez. Bu durumda, geliştiriciler uygulamanın beklenmeyen veya güvenlikle ilgili olayları güvenli şekilde ele almasını sağlamak için dile veya çerçeveye uygun desenler kullanmalıdır.

Referanslar

Daha fazla bilgi için:

- OWASP Web Security Testing Guide: Testing for Error Handling
- OWASP Authentication Cheat Sheet section about error messages
- OWASP Logging Cheat Sheet
- OWASP Application Logging Vocabulary Cheat Sheet

V17 WebRTC

Kontrol Amacı

Web Gerçek Zamanlı İletişim (WebRTC), modern uygulamalarda gerçek zamanlı ses, görüntü ve veri alışverişini mümkün kılar. Kullanım yaygınlaştıkça, WebRTC altyapısının güvenliğinin sağlanması kritik hâle gelir. Bu bölüm, WebRTC sistemlerini geliştiren, barındıran veya entegre eden paydaşlar için güvenlik gereksinimlerini tanımlar.

WebRTC pazarı genel olarak üç kategoriye ayrılabilir:

1. Ürün Geliştiriciler: WebRTC ürün ve çözümlerini geliştiren ve sağlayan özel veya açık kaynak tedarikçileri. Odak noktaları, başkaları tarafından da kullanılabilecek sağlam ve güvenli WebRTC teknolojileri geliştirmektir.
2. Hizmet Olarak İletişim Platformları (Communication Platforms as a Service - CPaaS): WebRTC işlevlerini mümkün kılmak için API, SDK ve altyapı/platform sağlayan hizmet sağlayıcılarıdır. CPaaS sağlayıcıları, birinci kategorideki ürünleri kullanabilir veya kendi WebRTC yazılımlarını geliştirerek bu hizmetleri sunabilirler.
3. Hizmet Sağlayıcılar: Ürün geliştiricilerden veya CPaaS sağlayıcılarından alınan ürünleri kullanan ya da kendi WebRTC çözümlerini geliştiren kuruluşlardır. Bu sağlayıcılar çevrim içi konferans, sağlık, online eğitim gibi alanlarda gerçek zamanlı iletişimin kritik olduğu uygulamalar geliştirirler.

Bu bölümdeki güvenlik gereksinimleri, özellikle aşağıdaki gibi sistemleri kullanan Ürün Geliştiriciler, CPaaS'ler ve Hizmet Sağlayıcıları için geçerlidir:

- WebRTC uygulamalarını oluşturmak için açık kaynak çözümler kullananlar,
- Altyapılarının bir parçası olarak ticari WebRTC ürünlerini kullananlar,
- Kendi WebRTC çözümlerini geliştiren ya da çeşitli bileşenleri birleştirerek entegre hizmet sunanlar.

Not: Bu güvenlik gereksinimleri yalnızca CPaaS sağlayıcılarının sunduğu SDK ve API'leri doğrudan kullanan geliştiricileri kapsamaz. Bu tür durumlarda, güvenlikten büyük ölçüde CPaaS sağlayıcısı sorumlu olur ve ASVS gibi genel güvenlik standartları her zaman tam olarak geçerli olmayabilir.

V17.1 TURN Sunucusu

Bu bölüm, kendi TURN (Traversal Using Relays around NAT) sunucusunu işleten sistemler için güvenlik gereksinimlerini tanımlar. TURN sunucuları, kısıtlayıcı ağ ortamlarında medya iletimini sağlamak için kullanılır, ancak yanlış yapılandırıldıklarında güvenlik riski oluşturabilirler. Bu gereksinimler, güvenli IP filtrelemesi ve kaynak tüketimine karşı korumaları ele alır.

#	Açıklama	Seviye
17.1.1	TURN hizmetinin yalnızca özel amaçlar için ayrılmamış IP adreslerine (ör. iç ağlar, yayın adresleri, loopback) erişime izin verdiği doğrulanmalıdır. Bu gereksinim hem IPv4 hem de IPv6 adresleri için geçerlidir.	2
17.1.2	TURN hizmetinin, meşru kullanıcıların sunucuda çok sayıda bağlantı noktası açmaya çalışması durumunda kaynak tüketimi saldırılarına karşı dayanıklı olduğu doğrulanmalıdır.	3

V17.2 Medya

Bu gereksinimler yalnızca kendi WebRTC medya sunucularını (ör. SFU, MCU, kayıt sunucuları veya geçit sunucuları) barındıran sistemler için geçerlidir. Medya sunucuları, medya akışlarını işler ve dağıtır, bu da iletişim güvenliğini sağlamada kritik rol oynar. Medya akışlarını korumak dinlemeyi, kurcalamayı veya hizmet reddi (DoS) saldırılarını önlemek açısından önemlidir.

Özellikle, hız sınırlama, zaman damgalarını doğrulama, gerçek zamanlı aralıkları eşleştirmek için senkronize saatler kullanma ve taşmayı önlemek ve uygun zamanlamayı korumak için arabellekleri (buffer) yönetme gibi flood saldırılarına karşı korumalar uygulamak gerekir. Belirli bir medya oturumu için paketler çok hızlı gelirse, fazla paketler düşürülmelidir. Girdi doğrulaması uygulayarak, tamsayı taşmalarını (integer overflow) güvenli bir şekilde ele alarak, arabellek taşmalarını (buffer overflow) önleyerek ve diğer sağlam hata işleme tekniklerini kullanarak sistemi hatalı biçimlendirilmiş paketlerden korumak da önemlidir.

Sadece peer-to-peer medya iletişimi kullanan sistemler bu medya gereksinimlerinin kapsamı dışındadır.

DTLS (Datagram Transport Layer Security) kullanımıyla ilgili gereksinimler burada yer alır. Kriptografik anahtarların yönetimi için bir politika oluşturulması “Kriptografi”bölümünde ele alınır. Onaylı şifreleme yöntemleri için ASVS’nin Kriptografi Ek’i veya NIST SP 800-52 Rev. 2, BSI TR-02102-2 (2025-01) gibi belgeler referans alınabilir.

#	Açıklama	Seviye
17.2.1	DTLS sertifikası anahtarının, belgelenmiş kriptografik anahtar yönetim politikasına uygun şekilde yönetildiği ve korunduğu doğrulanmalıdır.	2
17.2.2	Medya sunucusunun onaylı DTLS şifre takımlarını (cipher suites) ve Güvenli Gerçek Zamanlı Aktarım Protokolü (Secure Real-time Transport Protocol - SRTP) için anahtarlar oluşturmaya yönelik DTLS Uzantısı için güvenli bir koruma profilini kullanacak ve destekleyecek şekilde yapılandırıldığı (DTLS-SRTP) doğrulanmalıdır.	2

#	Açıklama	Seviye
17.2.3	Real-time Transport Protocol (RTP) injection saldırılarının bir hizmet reddi durumuna veya medya akışlarına ses veya video medyası eklenmesine yol açmasını önlemek için medya sunucusunda Güvenli SRTP kimlik doğrulamasının kontrol edildiği doğrulanmalıdır.	2
17.2.4	Verify that the media server is able to continue processing incoming media traffic when encountering malformed Secure Real-time Transport Protocol (SRTP) packets.	2
17.2.5	Medya sunucusunun, meşru kullanıcılardan gelen SRTP flood'ı sırasında gelen medya trafiğini işlemeye devam edebildiği doğrulanmalıdır.	3
17.2.6	Medya sunucusunun DTLS içinde yer alan "ClientHello"yarış durumu güvenlik açığına karşı savunmasız olmadığı, medya sunucusunun güvenlik açığına sahip olduğunun açıkça bilinip bilinmediği kontrol edilerek veya yarış koşulu testi gerçekleştirilerek doğrulanmalıdır.	3
17.2.7	Medya sunucusuna bağlı ses veya video kayıt mekanizmalarının, meşru kullanıcı kaynaklı SRTP flood'ı sırasında medya trafiğini işlemeye devam edebildiği doğrulanmalıdır.	3
17.2.8	DTLS sertifikasının, SDP (Session Description Protocol) fingerprint özelliği ile karşılaştırıldığı ve doğrulama başarısız olursa medya akışının sonlandırıldığı, medya akışının özgünlüğünü sağlamak için doğrulanmalıdır.	3

V17.3 Sinyalizasyon

Bu bölüm, kendi WebRTC sinyalizasyon sunucularını işleten sistemler için güvenlik gereksinimlerini tanımlar. Sinyalizasyon, peer-to-peer iletişimi koordine eder ve oturum kurulumunu veya kontrolünü bozabilecek saldırılara karşı dayanıklı olmalıdır.

Sinyalizasyonun güvenli olması için sistemler bozuk girdileri zararsız biçimde işleyebilmeli ve yük altında erişilebilir kalmalıdır.

#	Açıklama	Seviye
17.3.1	Sinyalizasyon sunucusunun, flood saldırısı altında bile meşru sinyalizasyon mesajlarını işlemeye devam edebildiği doğrulanmalıdır. Bu, sinyalizasyon düzeyinde oran sınırlaması uygulanarak sağlanmalıdır.	2

#	Açıklama	Seviye
17.3.2	Sinyalizasyon sunucusunun, hizmet reddine yol açabilecek şekilde yapılandırılmış bozuk sinyalizasyon mesajlarıyla karşılaştığında bile meşru mesajları işlemeye devam edebildiği doğrulanmalıdır. Bu, girdi doğrulaması, integer overflow ve buffer overflow'a karşı savunma ve sağlam hata işleme teknikleri ile sağlanmalıdır.	2

Referanslar

Daha fazla bilgi için:

- WebRTC DTLS ClientHello DoS hakkında en iyi dokümantasyonlar: Enable Security's blog post aimed at security professionals ve ilgili, WebRTC geliştiricilerine yönelik white paper
- RFC 3550 - RTP: A Transport Protocol for Real-Time Applications
- RFC 3711 - The Secure Real-time Transport Protocol (SRTP)
- RFC 5764 - Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP))
- RFC 8825 - Overview: Real-Time Protocols for Browser-Based Applications
- RFC 8826 - Security Considerations for WebRTC
- RFC 8827 - WebRTC Security Architecture
- DTLS-SRTP Protection Profiles

Ek A: Sözlük

- **Absolute Maximum Session Lifetime (Mutlak Maksimum Oturum Ömrü)** –NIST tarafından "Genel Zaman Aşımı"olarak da adlandırılan bu süre, kullanıcı etkileşiminden bağımsız olarak kimlik doğrulamasının ardından bir oturumun etkin kalabileceği maksimum süredir. Bu, oturum sona erme süresinin bir bileşenidir.
- **Allowlist (İzin Listesi)** –İzin verilen veri veya işlemlerin listesi, örneğin girdi doğrulaması yapmak için izin verilen karakterlerin listesi.
- **Anti-forgery token** –Bir veya daha fazla belirtecin bir istekte geçirildiği ve isteğin beklenen bir uç noktadan geldiğinden emin olmak için uygulama sunucusu tarafından doğrulandığı bir mekanizmadır.
- **Application Security (Uygulama Güvenliği)** –Uygulama düzeyinde güvenlik, örneğin altta yatan işletim sistemine veya bağlı ağlara odaklanmak yerine Açık Sistemler Ara Bağlantı Referans Modeli'nin (Open Systems Interconnection Reference Model - OSI Model) uygulama katmanını oluşturan bileşenlerin analizine odaklanır.
- **Application Security Verification (Uygulama Güvenliği Doğrulaması)** –Bir uygulamanın OWASP ASVS'ye göre teknik değerlendirmesidir.

- **Application Security Verification Report (Uygulama Güvenliği Doğrulaması Raporu)** –Belirli bir uygulama için doğrulayıcı tarafından üretilen genel sonuçları ve destekleyici analizi belgeleyen bir rapordur.
- **Authentication (Kimlik Doğrulama)** –Bir uygulama kullanıcısının iddia ettiği kimliğinin doğrulanmasıdır.
- **Automated Verification (Otomatik Doğrulama)** –Sorunları bulmak için güvenlik açığı imzalarını kullanan otomatik araçların (dinamik analiz araçları, statik analiz araçları veya her ikisi) kullanılmasıdır.
- **Black box testing** –Bir uygulamanın iç yapılarına veya işleyişine bakmadan işlevselliğini inceleyen bir yazılım testi yöntemidir.
- **Common Weakness Enumeration** (CWE) –Yaygın yazılım güvenlik zayıflıklarının topluluk tarafından geliştirilen bir listesi. Ortak bir dil, yazılım güvenlik araçları için bir ölçüm kriteri ve zayıflık tanımlama, azaltma ve önleme çabaları için bir temel olarak hizmet eder.
- **Component (Bileşen)** –Diğer bileşenlerle iletişim kuran ilişkili disk ve ağ arayüzlerine sahip bağımsız bir kod birimidir.
- **Credential Service Provider (Kimlik Bilgisi Hizmet Sağlayıcısı)** (CSP) –Kimlik Sağlayıcısı (Identity Provider - IdP) olarak da adlandırılır. Diğer uygulamalar tarafından kimlik doğrulama kaynağı olarak kullanılabilecek bir kullanıcı verisi kaynağıdır.
- **Cross-Site Script Inclusion (Siteler Arası Komut Dosyası Ekleme)** (XSSI) - Bir web uygulamasının harici bir kaynaktan kötü amaçlı kod aldığı ve bu kodu kendi içeriğinin bir parçası olarak dahil ettiği Siteler Arası Komut Çalıştırma (XSS) saldırısının bir çeşididir.
- **Cross-Site Scripting (Siteler Arası Komut Çalıştırma)** (XSS) –Genellikle web uygulamalarında bulunan ve istemci tarafı komut dosyalarının içeriğe eklenmesine izin veren bir güvenlik açığıdır.
- **Cryptographic module (Kriptografik modül)** –Kriptografik algoritmalar uygulayan ve/veya kriptografik anahtarlar üreten donanım, yazılım ve/veya ürün yazılımıdır.
- **Cryptographically secure pseudo-random number generator (Kriptografik olarak güvenli sözde rastgele sayı üreteci)** (CSPRNG) - Kriptografide kullanıma uygun özelliklere sahip bir sözde rastgele sayı üreteci, kriptografik rastgele sayı üreteci (CRNG) olarak da adlandırılır.
- **Datagram Transport Layer Security (Datagram Taşıma Katmanı Güvenliği)** (DTLS) –Bir ağ bağlantısı üzerinden iletişim güvenliği sağlayan bir kriptografik protokoldür. TLS protokolüne dayanır ancak datagram odaklı protokolleri (genellikle UDP üzerinden) korumak için uyarlanmıştır. DTLS 1.3 için RFC 9147'de tanımlanmıştır.
- **Datagram Transport Layer Security Extension to Establish Keys for the Secure Real-time Transport Protocol (Güvenli Gerçek Zamanlı Aktarım Protokolü Anahtarlarını Oluşturmak için Datagram Aktarım Katmanı Güvenlik Uzantısı)** (DTLS-SRTP) –Bir SRTP oturumu için anahtar materyali oluşturmak üzere DTLS el sıkışmasını kullanmaya yönelik bir mekanizmadır. RFC 5764'te tanımlanmıştır.
- **Design Verification (Tasarım Doğrulama)** –Bir uygulamanın güvenlik mimarisinin teknik değerlendirmesidir.
- **Dynamic Application Security Testing (Dinamik Uygulama Güvenlik Testi)** (DAST) –Bir uygu-

lamadaki güvenlik açığını gösteren koşulları, uygulamanın çalışır durumunda tespit etmek için tasarlanan teknolojilerdir.

- **Dynamic Verification (Dinamik Doğrulama)** –Bir uygulamanın çalışması sırasında sorunları tespit etmek için güvenlik açığı imzalarını kullanan otomatik araçların kullanımıdır.
- **Fast IDentity Online** (FIDO) –Biyometri, Güvenilir Platform Modülü (Trusted Platform Modules - TPM), USB güvenlik token'ları gibi çeşitli kimlik doğrulama yöntemlerinin kullanımına olanak tanıyan bir kimlik doğrulama standartları kümesidir.
- **Hardware Security Module (Donanım Güvenlik Modülü)** (HSM) –Kriptografik anahtarları ve diğer gizli verileri korumalı şekilde saklayan donanım bileşenidir.
- **Hibernate Query Language** (HQL) –Hibernate ORM kütüphanesi tarafından kullanılan, görünüm olarak SQL'e benzeyen bir sorgu (query) dilidir.
- **HTTP Strict Transport Security (HTTP Sıkı Taşıma Güvenliği)** (HSTS) –Tarayıcının yalnızca TLS üzerinden ve geçerli bir sertifika ile yanıt dönen alan adına bağlanmasını zorunlu kılan bir ilkedir. `Strict-Transport-Security` header field ile etkinleştirilir.
- **HyperText Transfer Protocol** (HTTP) –Dağıtık, işbirlikçi, çok ortamlı bilgi sistemleri için bir uygulama protokolüdür. Dünya Çapında Ağ (WWW) veri iletişiminin temelini oluşturur.
- **SSL/TLS üzerinden HyperText Transfer Protocol** (HTTPS) –HTTP iletişimini Taşıma Katmanı Güvenliği (TLS) ile şifreleyerek güvenli hale getirme yöntemidir.
- **Identity Provider (Kimlik Sağlayıcı)** (IdP) –NIST belgelerinde Kimlik Bilgisi Hizmet Sağlayıcısı (CSP) olarak da adlandırılır. Diğer uygulamalar için kimlik doğrulama kaynağı sağlayan bir varlıktır.
- **Inactivity Timeout (Etkin Olmama Zaman Aşımı)** –Bir oturumun, kullanıcı uygulama ile etkileşime girmediği sürede aktif kalabileceği süredir. Oturum zaman aşımının bir bileşenidir.
- **Input Validation (Girdi Doğrulama)** –RFC 7519 tarafından tanımlanan ve bir başlık (doğrulama yöntemini belirtir), bir gövde (iddiaları içerir) ve bir imza (gövdeyi doğrulamak için kullanılır) olmak üzere üç bölümden oluşan bir JSON veri nesnesidir. Kendi kendini içeren bir token (self-contained token) türüdür.
- **Local File Inclusion** (LFI) - Bir uygulamadaki savunmasız dosya dahil etme işlemlerinin sömürülmesiyle, sunucuda hâlihazırda bulunan yerel dosyaların dahil edilmesine yol açan saldırı türüdür.
- **Malicious Code (Kötü Amaçlı Kod)** –Uygulamanın geliştirme aşamasında uygulama sahibinin bilgisi dışında eklenen ve uygulamanın tasarlanan güvenlik politikasını aşmayı amaçlayan koddur. Virüs veya solucan gibi kötü amaçlı yazılımlardan (malware) farklıdır!
- **Malware (Kötü Amaçlı Yazılım)** –Uygulama çalışırken, kullanıcı veya yönetici farkında olmadan uygulamaya enjekte edilen çalıştırılabilir koddur.
- **Message authentication code (Mesaj Doğrulama Kodu)** (MAC) - Verinin bütünlüğünü ve özgünlüğünü sağlamak amacıyla bir MAC üretim algoritmasıyla hesaplanan kriptografik sağlama (checksum) değeridir.
- **Multi-factor authentication (Çok Faktörlü Kimlik Doğrulama)** (MFA) –İki veya daha fazla tekil faktörün bir arada kullanıldığı kimlik doğrulama yöntemidir.
- **Mutual TLS** (mTLS) –Bkz. TLS istemci kimlik doğrulaması.

- **Object-relational Mapping** (ORM) –İlişkisel (tablolu) bir veritabanının, uygulama içinde nesne tabanlı bir model üzerinden referans alınmasını ve sorgulanmasını sağlayan sistemdir.
- **One-time Password (Tek Kullanımlık Parola)** (OTP) –Sadece tek bir kullanım için benzersiz olarak üretilen paroladır.
- **Open Worldwide Application Security Project** (OWASP) –Uygulama yazılımlarının güvenliğini geliştirmeye odaklı, küresel ve açık bir topluluktur. Misyonu, uygulama güvenliğini "görünür" kılmak ve bireyler ile kurumların güvenlik risklerine dair bilinçli kararlar verebilmesini sağlamaktır. Bkz: https://www.owasp.org/.
- **Password-Based Key Derivation Function 2** (PBKDF2) –Bir giriş metninden (örn. parola) ve rastgele bir tuz (salt) değerinden güçlü bir kriptografik anahtar türeten, tek yönlü özel bir algoritmadır. Sonuç değeri özgün parola yerine saklandığında, parolanın çevrimdışı ortamda kırılmasını zorlaştırmak amacıyla kullanılır.
- **Public Key Infrastructure (Açık Anahtar Altyapısı)** (PKI) –Açık anahtarların ilgili varlıkların kimlikleriyle ilişkilendirilmesini sağlayan bir yapıdır. Bu ilişkilendirme, bir sertifika otoritesi (CA) tarafından gerçekleştirilen kayıt ve sertifika oluşturma süreciyle sağlanır.
- **Public Switched Telephone Network (Genel Anahtarlamalı Telefon Ağı)** (PSTN) –Sabit hatlı ve mobil telefonları içeren geleneksel telefon ağıdır.
- **Real-time Transport Protocol (Gerçek Zamanlı Taşıma Protokolü)** (RTP) ve **Real-time Transport Control Protocol (Gerçek Zamanlı Taşıma Denetim Protokolü)** (RTCP) –Multimedya akışlarını taşımak için birlikte kullanılan iki protokoldür. WebRTC yığını tarafından kullanılır. RFC 3550'de tanımlanmıştır.
- **Reference Token** –Sunucuda tutulan duruma veya meta veriye işaret eden veya onu tanımlayan, bazen rastgele veya opak token olarak da adlandırılan bir token türüdür. Kendi kendini içeren token'lardan farklı olarak, içeriğinde anlamlı bilgi taşımaz ve bağlam tamamen sunucu tarafından sağlanır. Genellikle bir oturum tanımlayıcısı içerir veya doğrudan bu tanımlayıcıyı temsil eder.
- **Relying Party** (RP) –Genellikle, kullanıcının ayrı bir kimlik sağlayıcı ile kimlik doğrulaması yaptığına güvenen bir uygulamadır. Bu uygulama, kullanıcıya ait olduğunu doğrulamak için kimlik sağlayıcı tarafından sağlanan bir token'a veya imzalı beyanlara dayanır.
- **Remote File Inclusion** (RFI) - Uygulamadaki zafiyet içeren dosya dahil etme işlemlerinin sömürülmesiyle, uzak sistemlerden dosya dahil edilmesine yol açan saldırı türüdür.
- **Scalable Vector Graphics** (SVG) –İki boyutlu vektör grafiklerini tanımlamak için kullanılan, XML tabanlı bir işaretleme dilidir.
- **Secure Real-time Transport Protocol (Güvenli Gerçek Zamanlı Taşıma Protokolü)** (SRTP) and **Secure Real-time Transport Control Protocol (Güvenli Gerçek Zamanlı Taşıma Denetim Protokolü)** (SRTCP) –RTP ve RTCP protokollerinin, mesaj şifreleme, kimlik doğrulama ve bütünlük koruması desteği sunan güvenli sürümleridir. RFC 3711'de tanımlanmıştır.
- **Security Architecture (Güvenlik Mimarisi)** –Uygulamanın tasarımına dair bir soyutlamadır. Güvenlik kontrollerinin nerede ve nasıl kullanıldığını, kullanıcı ve uygulama verilerinin nerede bulunduğunu ve hassasiyet seviyelerini tanımlar.
- **Security Assertion Markup Language** (SAML) –Kimlik sağlayıcı ile güvenen taraf arasında

imzalı beyanların (genellikle XML nesneleri) iletilmesine dayalı, açık bir tek oturum açma (SSO) standardıdır.

- **Security Configuration (Güvenlik Yapılandırması)** –Uygulamanın çalışma zamanında, güvenlik kontrollerinin nasıl kullanılacağını etkileyen yapılandırma bilgileridir.
- **Security Control (Güvenlik Kontrolü)** –Bir güvenlik denetimi gerçekleştiren (örn. yetkilendirme kontrolü) ya da çağrıldığında güvenlikle ilgili bir etki oluşturan (örn. denetim kaydı üretme) işlev veya bileşendir.
- **Security information and event management** (SIEM) - Bir kuruluşun BT altyapısından gelen güvenlik verilerini toplayarak tehdit tespiti, uyumluluk ve olay yönetimi amacıyla analiz eden bir sistemdir.
- **Self-Contained Token (Kendi İçinde Taşıyıcı Token)** –Herhangi bir sunucu tarafı duruma veya harici saklamaya bağımlı olmayan, bir veya birden fazla niteliği içinde barındıran token'dır. Bu token'lar, içeriklerinin doğruluğunu ve bütünlüğünü sağlamak için dijital imza veya ileti doğrulama kodu (MAC) gibi kriptografik tekniklerle korunur. Örnekler: SAML beyanları, JWT.
- **Server-side Request Forgery** (SSRF) –Sunucudaki işlevselliğin suistimal edilerek iç kaynakların okunması veya güncellenmesi ile sonuçlanan saldırı türüdür. Saldırgan, sunucu üzerinde çalışacak olan URL'yi doğrudan sağlar ya da değiştirir.
- **Session Description Protocol** (SDP) –Multimedya oturumu kurulumunda kullanılan bir mesaj formatıdır (örneğin WebRTC'de kullanılır). RFC 4566'da tanımlanmıştır.
- **Session Identifier** veya **Session ID** –Backend'de saklanan durumlu bir oturumu tanımlayan anahtardır. İstemciye, doğrudan veya bir Referans Token içinde aktarılır.
- **Session Token** –Bu standartta, self-contained token kullanan ve durum barındırmayan oturum mekanizmalarında ya da reference token kullanan ve durum barındıran mekanizmalarda kullanılan token veya değeri ifade eden genel bir terimdir.
- **Session Traversal Utilities for NAT** (STUN) –Eşler arası iletişim kurmak amacıyla NAT geçişine yardımcı olan bir protokoldür. RFC 3489'da tanımlanmıştır.
- **Single-factor authenticator (Tek Faktörlü Kimlik Doğrulayıcı)** –Kullanıcının kimliğini doğrulamak için kullanılan mekanizmadır. Bu mekanizma ya bilinen bir şey (parolalar, PIN'ler), ya sahip olunan bir şey (OTP token'ları, akıllı kart gibi kriptografik cihazlar), ya da kullanıcıya özgü bir şey (biyometrik veriler, parmak izi, yüz taraması) olmalıdır.
- **Single Sign-on Authentication (Tek Oturum Açma)** (SSO) –Kullanıcının bir uygulamaya giriş yaptıktan sonra, ek bir kimlik doğrulama gerekmeksizin diğer uygulamalara da otomatik olarak giriş yapabilmesidir. Örneğin, Google hesabına giriş yapan bir kullanıcı, YouTube, Google Dokümanlar ve Gmail'e de otomatik olarak erişim sağlar.
- **Software bill of materials (Yazılım Bileşen Listesi)** (SBOM) - Bir yazılım uygulamasını oluşturmak veya birleştirmek için gerekli tüm bileşenlerin, modüllerin, kütüphanelerin, çatıların (framework) ve diğer kaynakların yapılandırılmış ve kapsamlı bir listesidir.
- **Software Composition Analysis** (SCA) –Uygulamanın bileşimini, bağımlılıklarını, kütüphanelerini ve paketlerini analiz ederek kullanılan belirli bileşen sürümlerindeki güvenlik açıklarını tespit etmeye yönelik teknolojiler bütünüdür. Kaynak kod analizi (SAST) ile karıştırılmamalıdır.
- **Software development lifecycle** (SDLC) –Yazılımın, ilk gereksinimlerden dağıtım ve bakıma

kadar geçen süreçte adım adım geliştirilmesini tanımlayan modeldir.

- **SQL Injection** (SQLi) –Veri odaklı uygulamalara yönelik bir kod enjeksiyon tekniğidir. Kötü niyetli SQL ifadeleri bir giriş noktasına enjekte edilir.
- **Stateful Session Mechanism** –Oturum durumunun backend'de tutulduğu oturum yönetim biçimidir. Genellikle kriptografik olarak güvenli sahte rastgele sayı üreteci (CSPRNG) ile oluşturulmuş bir oturum token'ı son kullanıcıya verilir.
- **Stateless Session Mechanism** –Hizmet tarafından saklanmayan, oturum bilgilerini taşıyan kendi kendini içeren bir token'ın istemcilere iletildiği mekanizmadır. Gerçekte, hizmetin belirli güvenlik kontrollerini uygulayabilmesi için (örneğin JWT iptal listesi gibi) oturum bilgilerine kısmen erişmesi gerekebilir.
- **Static application security testing (Statik Uygulama Güvenlik Testi)** (SAST) –Uygulamanın kaynak kodunu, bayt kodunu ve ikili dosyalarını çalışmayan (non-running) durumda analiz ederek güvenlik açıklarını ortaya çıkaran teknolojiler bütünüdür. Uygulamayı "içeriden dışarı" analiz eder.
- **Threat Modeling (Tehdit Modelleme)** –Tehdit aktörleri, güvenlik bölgeleri, güvenlik kontrolleri ile teknik ve iş varlıklarını tanımlamak amacıyla gittikçe rafine edilen güvenlik mimarilerinin geliştirilmesini içeren bir tekniktir.
- **Time-of-check to time-of-use (Denetim Zamanı ile Kullanım Zamanı Arasındaki Süre)** (TOCTOU) –Bir kaynağın durumu uygulama tarafından kullanılmadan önce kontrol edilir, fakat bu durum kontrol ile kullanım arasında değişebilir. Bu da kontrolün geçersiz hale gelmesine ve yanlış işlemlerin gerçekleştirilmesine neden olabilir.
- **Time based One-time Passwords (Zamana Dayalı Tek Kullanımlık Parola)** (TOTPs) - Şifreyi üretmek için algoritmada mevcut zamanın kullanıldığı bir OTP üretim yöntemidir.
- **TLS client authentication (TLS İstemci Kimlik Doğrulaması)** veya **Mutual TLS (Karşılıklı TLS)** (mTLS) –Standart bir TLS bağlantısında istemci, sunucunun kimliğini sunucu sertifikası ile doğrular. TLS istemci kimlik doğrulamasında ise istemci de kendi özel anahtarı ve sertifikası ile kendi kimliğini sunucuya doğrulatır.
- **Transport Layer Security (Taşıma Katmanı Güvenliği)** (TLS) –Ağ üzerinden güvenli iletişim sağlamak için kullanılan kriptografik protokoller bütünüdür.
- **Traversal Using Relays around NAT (NAT Etrafında Aktarıcılar Kullanılarak Geçiş)** (TURN) –Eşler arası doğrudan bağlantılar kurulamadığında, bir TURN sunucusu aracılığıyla veri iletimi yapılmasını sağlayan STUN protokolünün bir uzantısıdır. RFC 8656'da tanımlanmıştır.
- **Trusted execution environment (Güvenli Yürütme Ortamı)** (TEE) - Sistemin geri kalanından izole bir işlem ortamında uygulamaların güvenli bir şekilde çalıştırılabildiği özel bir alan.
- **Trusted Platform Module (Güvenilir Platform Modülü)** (TPM) –Genellikle anakart gibi daha büyük donanımlara bağlı olan, sistemin "güven kökü (root of trust)"olarak görev yapan HSM türüdür.
- **Trusted Service Layer (Güvenilir Hizmet Katmanı)** –Mikro servis, sunucusuz API, sunucu tarafı, güvenli önyüklemeli istemci API'si, ortak veya harici API'ler gibi, bir güvenlik kontrolü uygulayan ve kötü niyetli kullanıcıların atlayamayacağı katmanlardır.
- **Uniform Resource Identifier** (URI)- Web sayfası, e-posta adresi veya konum gibi bir kaynağı

tanımlayan benzersiz karakter dizisidir.
- **Uniform Resource Locator** (URL) –İnternette bir kaynağın konumunu belirten karakter dizisidir.
- **Universally Unique Identifier (Evrensel Benzersiz Tanımlayıcı)** (UUID) –Yazılımlarda kimlik belirleme amacıyla kullanılan benzersiz referans numarasıdır.
- **Verifier (Doğrulayıcı)** –Bir uygulamayı OWASP ASVS gereksinimlerine göre inceleyen kişi veya ekip.
- **Web Real-Time Communication** (WebRTC) –Web uygulamalarında genellikle görüntülü görüşme amacıyla kullanılan, multimedya akışlarını taşıyan protokol yığını ve web API'sidir. SRTP, SRTCP, DTLS, SDP ve STUN/TURN üzerine kuruludur.
- **WebSocket over TLS** (WSS) –WebSocket iletişimini TLS protokolü ile katmanlayarak güvenli hale getirme uygulamasıdır.
- **What You See Is What You Get** (WYSIWYG) –İçeriğin nasıl görüntüleneceğini doğrudan düzenleme sırasında gösteren zengin metin düzenleyicisidir. Kod görünümü yerine gerçek görünümü gösterir.
- **X.509 Certificate** –Uluslararası X.509 açık anahtar altyapısı (PKI) standardını kullanan ve bir açık anahtarın sertifikada yer alan kullanıcıya, bilgisayara veya hizmete ait olduğunu doğrulayan dijital sertifikadır.
- **XML eXternal Entity** (XXE) –Sistem tanımlayıcısı ile yerel veya uzak içeriğe erişebilen XML varlığı türüdür. Farklı enjeksiyon saldırılarına neden olabilir.

Ek B: Referanslar

Aşağıdaki OWASP projeleri, bu standardın kullanıcıları/uygulayıcıları için yararlı olabilir:

OWASP Ana Projeleri

1. OWASP Top 10 Project: https://owasp.org/www-project-top-ten/
2. OWASP Web Security Testing Guide: https://owasp.org/www-project-web-security-testing-guide/
3. OWASP Proactive Controls: https://owasp.org/www-project-proactive-controls/
4. OWASP Software Assurance Maturity Model (SAMM): https://owasp.org/www-project-samm/
5. OWASP Secure Headers Project: https://owasp.org/www-project-secure-headers/

OWASP Cheat Sheet Series Projesi

Bu proje ASVS'deki farklı konularla ilgili olacak birkaç cheat sheet'e sahiptir.

ASVS ile ilgili bir haritalandırmaya buradan ulaşabilirsiniz: https://cheatsheetseries.owasp.org/IndexASVS.html

Mobil Güvenlikle İlgili Projeler

1. OWASP Mobile Security Project: https://owasp.org/www-project-mobile-security/
2. OWASP Mobile Top 10 Risks: https://owasp.org/www-project-mobile-top-10/
3. OWASP Mobile Security Testing Guide and Mobile Application Security Verification Standard: https://owasp.org/www-project-mobile-security-testing-guide/

OWASP Nesnelerin İnterneti (Internet of Things) İle İlgili Projeler

1. OWASP Internet of Things Project: https://owasp.org/www-project-internet-of-things/

OWASP Serverless Projeleri

1. OWASP Serverless Project: https://owasp.org/www-project-serverless-top-10/

Diğerleri

Benzer şekilde, aşağıdaki web siteleri de bu standardın kullanıcıları/uygulayıcıları için yararlı olabilir:

1. SecLists Github: https://github.com/danielmiessler/SecLists
2. MITRE Common Weakness Enumeration: https://cwe.mitre.org/
3. PCI Security Standards Council: https://www.pcisecuritystandards.org/
4. PCI Data Security Standard (DSS) v3.2.1 Requirements and Security Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf
5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf
6. PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf
7. OWASP ASVS 4.0 Testing Guide https://github.com/BlazingWind/OWASP-ASVS-4.0-testing-guide

Ek C: Kriptografi Standartları

"Kriptografi"bölümü sadece en iyi uygulamaları tanımlamanın ötesine geçmektedir. Kriptografi ilkelerinin daha iyi anlaşılmasını ve daha esnek, modern güvenlik yöntemlerinin benimsenmesini teşvik etmeyi amaçlamaktadır. Bu ek, "Kriptografi"bölümünde ana hatlarıyla belirtilen kapsayıcı standartları tamamlayan her bir gereklilikle ilgili ayrıntılı teknik bilgi sağlamaktadır.

Bu ek, farklı kriptografik mekanizmalar için onay seviyesini tanımlar:

- Onaylanmış (A) mekanizmalar uygulamalarda kullanılabilir.

- Eski (Legacy) mekanizmalar (L) uygulamalarda kullanılmamalıdır, ancak yine de yalnızca mevcut eski uygulamalar veya kodlarla uyumluluk için kullanılabilir. Bu tür mekanizmaların kullanımı şu anda kendi başına bir güvenlik açığı olarak kabul edilmese de, mümkün olan en kısa sürede daha güvenli ve geleceğe dönük mekanizmalarla değiştirilmelidir.
- İzin verilmeyen (Disallowed) mekanizmalar (D) şu anda bozuk kabul edildikleri veya yeterli güvenlik sağlamadıkları için kullanılmamalıdır.

Bu liste, belirli bir uygulama bağlamında, aşağıdakiler de dahil olmak üzere çeşitli nedenlerle geçersiz kılınabilir:

- kriptografi alanındaki yeni gelişmeler;
- yönetmeliklere uygunluk.

Kriptografik Envanter ve Dokümantasyon

Bu bölüm V11.1 Kriptografik Envanter ve Dokümantasyon için ek bilgi sağlamaktadır.

Algoritmalar, anahtarlar ve sertifikalar gibi tüm kriptografik varlıkların düzenli olarak keşfedildiğinden, envanterinin çıkarıldığından ve değerlendirildiğinden emin olmak önemlidir. Seviye 3 için bu, bir uygulamada kriptografi kullanımını keşfetmek için statik ve dinamik tarama kullanımını içermelidir. SAST ve DAST gibi araçlar bu konuda yardımcı olabilir, ancak daha geniş bir kapsam elde etmek için özel araçlara ihtiyaç duyulması mümkündür. Ücretsiz araç örnekleri şunları içerir:

- CryptoMon - Network Cryptography Monitor - using eBPF, written in python
- Cryptobom Forge Tool: Generating Comprehensive CBOMs from CodeQL Outputs

Kriptografik Parametrelerin Eşdeğer Güçleri

Çeşitli kriptografik sistemler için göreceli güvenlik güçleri bu tabloda yer almaktadır: NIST SP 800-57 Kısım 1, syf.71

Güvenlik Gücü	Simetrik Anahtar Algoritmaları	Sonlu Alan	Tamsayı Çarpanlarına Ayırma	Eliptik Eğri (Elliptic Curve)
<= 80	2TDEA	L = 1024 N = 160	k = 1024	f = 160-223
112	3TDEA	L = 2048 N = 224	k = 2048	f = 224-255
128	AES-128	L = 3072 N = 256	k = 3072	f = 256-383
192	AES-192	L = 7680 N = 384	k = 7680	f = 384-511
256	AES-256	L = 15360 N = 512	k = 15360	f = 512+

Uygulama örnekleri şunlardır:

- Sonlu Alan Kriptografisi (Finite Field Cryptography): DSA, FFDH, MQV
- Tamsayı Çarpanlarına Ayırma Kriptografisi (Integer Factorization Cryptography): RSA
- Eliptik Eğri Kriptografisi (Elliptic Curve Cryptography): ECDSA, EdDSA, ECDH, MQV

Not: Bu bölümde kuantum bilgisayarlarının var olmadığı varsayılmaktadır. Eğer böyle bir bilgisayar var olsaydı, son 3 sütun için yapılan tahminler artık geçerli olmazdı.

Rastgele Değerler

Bu bölümde V11.5 Rastgele Değerler için ek bilgi sağlanmaktadır.

İsim	Sürüm/Referans	Notlar	Durum
`/dev/random`	Linux 4.8+ (Oct 2016), , iOS, Android ve diğer Linux tabanlı POSIX işletim sistemlerinde de bulunur. RFC7539 temel alınmıştır.	ChaCha20 akışını kullanır. iOS `SecRandomCopyBytes` ve Android `Secure Random` içinde her biri için sağlanan doğru ayarlarla bulunur.	A
`/dev/urandom`	Linux çekirdeğinin rastgele veri sağlamak için özel dosyası	Donanım rastgeleliğinden gelen yüksek kaliteli, entropi kaynakları sağlar	A
`AES-CTR-DRBG`	NIST SP800-90A	Windows CNG API `BCryptGenRandom` gibi yaygın uygulamalarda kullanıldığı gibi, `BCRYPT_RNG_ALGORITHM` tarafından ayarlanır.	A
`HMAC-DRBG`	NIST SP800-90A		A
`Hash-DRBG`	NIST SP800-90A		A
`getentropy()`	OpenBSD, Linux glibc 2.25+ ve macOS 10.12+ sürümlerinde mevcuttur.	Doğrudan çekirdeğin entropi kaynağından basit ve minimal bir API ile güvenli rastgele baytlar sağlar. Daha moderndir ve eski API'lerle ilişkili tuzaklardan kaçınır.	A

HMAC-DRBG veya Hash-DRBG ile kullanılan temel hash fonksiyonu, bu kullanım için onaylanmış olmalıdır.

Şifre (Cipher) Algoritmaları

Bu bölüm V11.3 Şifreleme Algoritmaları için ek bilgi sağlar.

Onaylanan şifre algoritmaları tercih sırasına göre listelenmiştir.

Simetrik Anahtar Algoritmaları	Referans	Durum
AES-256	FIPS 197	A
Salsa20	Salsa 20 specification	A
XChaCha20	XChaCha20 Draft	A
XSalsa20	Extending the Salsa20 nonce	A
ChaCha20	RFC 8439	A
AES-192	FIPS 197	A
AES-128	FIPS 197	L
2TDEA		D
TDEA (3DES/3DEA)		D
IDEA		D
RC4		D
Blowfish		D
ARC4		D
DES		D

AES Şifre Modları

AES gibi blok şifreler farklı işlem modları ile kullanılabilir. Elektronik kod kitabı (Electronic codebook - ECB) gibi birçok işlem modu güvensizdir ve kullanılmamalıdır. Galois/Counter Modu (Galois/Counter Mode - GCM) ve şifre blok zincirleme mesaj kimlik doğrulama kodlu Sayaç (CCM) işlem modları, kimliği doğrulanmış şifreleme sağlar ve modern uygulamalarda kullanılmalıdır.

Onaylanan modlar tercih sırasına göre listelenmiştir.

Mod	Kimliği Doğrulanmış	Referans	Durum	Kısıtlama
GCM	Evet	NIST SP 800-38D	A	
CCM	Evet	NIST SP 800-38C	A	
CBC	Hayır	NIST SP 800-38A	L	

Mod	Kimliği Doğrulanmış	Referans	Durum	Kısıtlama
CCM-8	Evet		D	
ECB	Hayır		D	
CFB	Hayır		D	
OFB	Hayır		D	
CTR	Hayır		D	

Notlar:

- Tüm şifrelenmiş mesajların kimliği doğrulanmalıdır. CBC modunun herhangi bir kullanımı için, mesajı doğrulamak üzere ilişkili bir karma MAC algoritması olmalıdır. Genel olarak bu, Encrypt-Then-Hash yönteminde uygulanmalıdır (ancak TLS 1.2 bunun yerine Hash-Then-Encrypt kullanır). Eğer bu garanti edilemiyorsa, CBC kullanılmamalıdır. MAC algoritması olmadan şifrelemeye izin verilen tek uygulama disk şifrelemedir.
- Eğer CBC kullanıldıysa, padding doğrulanmasının sabit zamanda gerçekleştirildiği garanti edilmelidir.
- CCM-8 kullanıldığında, MAC etiketi yalnızca 64 bit güvenliğe sahiptir. Bu, en az 128 bit güvenlik gerektiren gereksinim 6.2.9 ile uyumlu değildir.
- Disk şifrelemenin ASVS için kapsam dışı olduğu düşünülmektedir. Bu nedenle bu ekte disk şifreleme için onaylanmış herhangi bir yöntem listelenmemiştir. Bu kullanım için, kimlik doğrulama olmadan şifreleme genellikle kabul edilir ve XTS, XEX ve LRW modları tipik olarak kullanılır.

Anahtar Sarma (Key Wrapping)

Kriptografik anahtar sarma (ve buna karşılık gelen anahtar açma), mevcut bir anahtarı ek bir şifreleme mekanizması kullanarak kapsülleyerek (yani sararak) koruma yöntemidir. Böylece orijinal anahtar, örneğin bir aktarım sırasında açıkça ortaya çıkmaz. Orijinal anahtarı korumak için kullanılan bu ek anahtar sarma anahtarı (wrap key) olarak adlandırılır.

Bu işlem, güvenilmez olduğu düşünülen yerlerdeki anahtarları korumak veya hassas anahtarları güvenilmeyen ağlar üzerinden ya da uygulamalar içinde göndermek istendiğinde gerçekleştirilebilir. Ancak, bir sarma/açma prosedürüne girişmeden önce orijinal anahtarın doğasını (örn. kimliği ve amacı) anlamaya ciddi önem verilmelidir, çünkü bunun hem kaynak hem de hedef sistemler/uygulamalar için güvenlik ve özellikle de bir anahtarın işlevinin (örn. imzalama) denetim izlerini ve uygun anahtar depolamayı içerebilecek uyumluluk açısından yansımaları olabilir.

Özellikle, NIST SP 800-38F uyarınca ve kuantum tehdidine karşı ileriye dönük hükümler göz önünde bulundurularak, anahtar sarma için AES-256 kullanılmalıdır. AES kullanan şifre modları tercih sırasına göre şunlardır:

Anahtar Sarma	Referans	Durum
KW	NIST SP 800-38F	A
KWP	NIST SP 800-38F	A

Kullanım durumu gerektiriyorsa AES-192 ve AES-128 kullanılabilir. Ancak bunun sebebi ve amacı, kuruluşun kriptografi envanterinde belgelenmelidir.

Kimlik Doğrulamalı Şifreleme

Disk şifreleme haricinde, şifrelenmiş veriler bir çeşit kimliği doğrulanmış şifreleme (authenticated encryption - AE) şeması kullanılarak, genellikle ilişkili verilerle kimliği doğrulanmış şifreleme (authenticated encryption with associated data - AEAD) şeması kullanılarak yetkisiz değişikliklere karşı korunmalıdır.

Uygulama tercihen onaylı bir AEAD şeması kullanmalıdır. Alternatif olarak onaylı bir şifreleme şeması ile onaylı bir MAC algoritmasını bir Encrypt-then-MAC yapısıyla birleştirebilir.

MAC-then-encrypt'e eski uygulamalarla uyumluluk için hala izin verilmektedir. TLS v1.2'de eski şifre takımları ile kullanılır.

AEAD mekanizması	Referans	Durum
AES-GCM	SP 800-38D	A
AES-CCM	SP 800-38C	A
ChaCha-Poly1305	RFC 7539	A
AEGIS-256	AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)	A
AEGIS-128	AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)	A
AEGIS-128L	AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)	A
Encrypt-then-MAC		A
MAC-then-encrypt		L

Hash Fonksiyonları

Bu bölümde V11.4 Hashing ve Hash Tabanlı Fonksiyonlar için ek bilgiler sağlanmaktadır.

Genel Kullanım Durumları için Hash Fonksiyonları

Aşağıdaki tabloda, dijital imzalar gibi genel kriptografik kullanım durumlarında onaylanan hash fonksiyonları listelenmektedir:

- Onaylanan hash fonksiyonları güçlü çakışma (collision) direnci sağlar ve yüksek güvenlikli uygulamalar için uygundur.
- Bu algoritmalardan bazıları uygun kriptografik anahtar yönetimi ile kullanıldığında saldırılara karşı güçlü direnç sunar ve bu nedenle HMAC, KDF ve RBG işlevleri için ek olarak onaylanmıştır.
- 254 bitten daha az çıktıya sahip hash fonksiyonların çakışma direnci yetersizdir ve dijital imza veya çakışma direnci gerektiren diğer uygulamalar için kullanılmamalıdır. Diğer kullanımlar için, sadece eski sistemlerle uyumluluk ve doğrulama için kullanılabilirler, ancak yeni tasarımlarda kullanılmamalıdırlar.

Hash fonksiyonu	Referans	Durum	Kısıtlamalar
SHA3-512	FIPS 202	A	
SHA-512	FIPS 180-4	A	
SHA3-384	FIPS 202	A	
SHA-384	FIPS 180-4	A	
SHA3-256	FIPS 202	A	
SHA-512/256	FIPS 180-4	A	
SHA-256	FIPS 180-4	A	
SHAKE256	FIPS 202	A	
BLAKE2s	BLAKE2: simpler, smaller, fast as MD5	A	
BLAKE2b	BLAKE2: simpler, smaller, fast as MD5	A	
BLAKE3	BLAKE3 one function, fast everywhere	A	
SHA-224	FIPS 180-4	L	HMAC, KDF, RBG, dijital imzalar için uygun değildir.
SHA-512/224	FIPS 180-4	L	HMAC, KDF, RBG, dijital imzalar için uygun değildir.
SHA3-224	FIPS 202	L	HMAC, KDF, RBG, dijital imzalar için uygun değildir.

Hash fonksiyonu	Referans	Durum	Kısıtlamalar
SHA-1	RFC 3174 & RFC 6194	L	HMAC, KDF, RBG, dijital imzalar için uygun değildir.
CRC (herhangi bir uzunluk)		D	
MD4	RFC 1320	D	
MD5	RFC 1321	D	

Parola Saklama için Hash Fonksiyonları

Güvenli parola hashleme için özel hash fonksiyonları kullanılmalıdır. Bu yavaş hash algoritmaları, parola kırmanın hesaplama zorluğunu artırarak kaba kuvvet ve sözlük saldırılarını azaltır.

KDF	Referans	Gerekli Parametreler	Durum
argon2id	RFC 9106	t = 1: m ≥ 47104 (46 MiB), p = 1	A
		t = 2: m ≥ 19456 (19 MiB), p = 1	A
		t ≥ 3: m ≥ 12288 (12 MiB), p = 1	A
scrypt	RFC 7914	p = 1: N ≥ 2^17 (128 MiB), r = 8	A
		p = 2: N ≥ 2^16 (64 MiB), r = 8	A
		p ≥ 3: N ≥ 2^15 (32 MiB), r = 8	A
bcrypt	A Future-Adaptable Password Scheme	cost ≥ 10	A
PBKDF2-HMAC-SHA-512	NIST SP 800-132, FIPS 180-4	iterations ≥ 210,000	A
PBKDF2-HMAC-SHA-256	NIST SP 800-132, FIPS 180-4	iterations ≥ 600,000	A
PBKDF2-HMAC-SHA-1	NIST SP 800-132, FIPS 180-4	iterations ≥ 1,300,000	L

Parola depolama için onaylı parola tabanlı anahtar türetme işlevleri kullanılabilir.

Anahtar Türetme İşlevleri (Key Derivation Functions - KDF)

Genel Anahtar Türetme Fonksiyonları

KDF	Referanslar	Durum
HKDF	RFC 5869	A
TLS 1.2 PRF	RFC 5248	L
MD5-based KDFs	RFC 1321	D
SHA-1-based KDFs	RFC 3174 & RFC 6194	D

Parola Tabanlı Anahtar Türetme İşlevleri

KDF	Referans	Gerekli Parametreler	Durum
argon2id	RFC 9106	t = 1: m ≥ 47104 (46 MiB), p = 1	A
		t = 2: m ≥ 19456 (19 MiB), p = 1	A
		t ≥ 3: m ≥ 12288 (12 MiB), p = 1	A
scrypt	RFC 7914	p = 1: N ≥ 2^17 (128 MiB), r = 8	A
		p = 2: N ≥ 2^16 (64 MiB), r = 8	A
		p ≥ 3: N ≥ 2^15 (32 MiB), r = 8	A
PBKDF2-HMAC-SHA-512	NIST SP 800-132, FIPS 180-4	iterations ≥ 210,000	A
PBKDF2-HMAC-SHA-256	NIST SP 800-132, FIPS 180-4	iterations ≥ 600,000	A
PBKDF2-HMAC-SHA-1	NIST SP 800-132, FIPS 180-4	iterations ≥ 1,300,000	L

Anahtar Değişim Mekanizmaları (Key Exchange Mechanisms - KEX)

Bu bölümde V11.6 Açık Anahtar Kriptografisi için ek bilgi sağlanmaktadır.

KEX Şemaları

Tüm anahtar değişimi şemaları için 112 bit veya üzeri bir güvenlik gücü sağlanmalıdır ve bunların uygulanması aşağıdaki tablodaki parametre seçeneklerini takip etmelidir.

Şema	Etki Alanı Parametreleri	İleriye Yönelik Gizlilik	Durum
Finite Field Diffie-Hellman (FFDH)	L >= 3072 & N >= 256	Evet	A
Elliptic Curve Diffie-Hellman (ECDH)	f >= 256-383	Evet	A
Encrypted key transport with RSA-PKCS#1 v1.5		Hayır	D

Parametreler aşağıdaki gibidir:

- k, RSA anahtarları için anahtar boyutudur.
- L açık anahtarın boyutu ve N sonlu alan kriptografisi için özel anahtarın boyutudur.
- f, ECC için anahtar boyutları aralığıdır.

Herhangi bir yeni uygulama NIST SP 800-56A & B ve NIST SP 800-77 ile uyumlu OLMAYAN herhangi bir şema kullanılmamalıdır. Özellikle, IKEv1 üretimde kullanılmamalıdır.

Diffie-Hellman Grupları

Aşağıdaki gruplar Diffie-Hellman anahtar değişimi uygulamaları için onaylanmıştır. Güvenlik güçleri NIST SP 800-56A, Ek D ve NIST SP 800-57 Bölüm 1 Rev.5'de belgelenmiştir.

Group	Status
P-224, secp224r1	A
P-256, secp256r1	A
P-384, secp384r1	A
P-521, secp521r1	A
K-233, sect233k1	A
K-283, sect283k1	A
K-409, sect409k1	A
K-571, sect571k1	A
B-233, sect233r1	A
B-283, sect283r1	A
B-409, sect409r1	A
B-571, sect571r1	A
Curve448	A
Curve25519	A

Group	Status
MODP-2048	A
MODP-3072	A
MODP-4096	A
MODP-6144	A
MODP-8192	A
ffdhe2048	A
ffdhe3072	A
ffdhe4096	A
ffdhe6144	A
ffdhe8192	A

Mesaj Kimlik Doğrulama Kodları (Message Authentication Codes - MAC)

Mesaj Kimlik Doğrulama Kodları (MAC), bir mesajın bütünlüğünü ve gerçekliğini doğrulamak için kullanılan kriptografik yapılardır. Bir MAC, girdi olarak bir mesaj ve gizli bir anahtar alır ve sabit boyutlu bir etiket (MAC değeri) üretir. MAC'ler güvenli iletişim protokollerinde (örneğin TLS/SSL) taraflar arasında değiş tokuş edilen mesajların gerçek ve bozulmamış olmasını sağlamak için yaygın olarak kullanılır.

MAC Algoritması	Referans	Durum
HMAC-SHA-256	RFC 2104 & FIPS 198-1	A
HMAC-SHA-384	RFC 2104 & FIPS 198-1	A
HMAC-SHA-512	RFC 2104 & FIPS 198-1	A
KMAC128	NIST SP 800-185	A
KMAC256	NIST SP 800-185	A
BLAKE3 (keyed_hash mode)	BLAKE3 one function, fast everywhere	A
AES-CMAC	RFC 4493 & NIST SP 800-38B	A
AES-GMAC	NIST SP 800-38D	A
Poly1305-AES	The Poly1305-AES message-authentication code	A
HMAC-SHA-1	RFC 2104 & FIPS 198-1	L
HMAC-MD5	RFC 1321	D

Dijital İmzalar

İmza şemaları NIST SP 800-57 Bölüm 1 uyarınca onaylanmış anahtar boyutlarını ve parametrelerini kullanmalıdır.

İmza Algoritması	Referans	Durum
EdDSA (Ed25519, Ed448)	RFC 8032	A
XEdDSA (Curve25519, Curve448)	XEdDSA	A
ECDSA (P-256, P-384, P-521)	FIPS 186-4	A
RSA-RSSA-PSS	RFC 8017	A
RSA-SSA-PKCS#1 v1.5	RFC 8017	D
DSA (any key size)	FIPS 186-4	D

Kuantum Sonrası Şifreleme Standartları

Kuantum sonrası kriptografi (PQC) uygulamaları FIPS-203, FIPS-204 ve FIPS-205 standartlarını takip etmelidir. Şu anda, bu standartlar için çok fazla güçlendirilmiş kod örneği veya referans uygulaması mevcut değildir. Daha fazla ayrıntı için, Kuantum sonrası şifreleme standartlarının ilk üçüne ilişkin NIST duyurusu (Ağustos 2024) adresine bakın.

Önerilen mlkem768x25519 kuantum sonrası hibrit TLS anahtar anlaşma yöntemi Firefox sürüm 132 ve Chrome sürüm 131 gibi büyük tarayıcılar tarafından desteklenmektedir. Kriptografik test ortamlarında veya endüstri veya devlet onaylı kütüphanelerde mevcut olduğunda kullanılabilir.

Ek D: Tavsiyeler

Giriş

Uygulama Güvenliği Doğrulama Standardının (ASVS) 5.0 sürümü hazırlanırken, 5.0'da gereklilik olarak yer almaması gereken bir dizi mevcut ve yeni önerilen öğe olduğu anlaşıldı. Bunun nedeni, 5.0 tanımına göre ASVS kapsamında olmamaları veya alternatif olarak iyi bir fikir olmalarına rağmen zorunlu hale getirilemeyeceklerinin düşünülmesi olabilir.

Tüm bu maddeleri tamamen kaybetmek istemediğimiz için bazıları bu ekte yer almaktadır.

Önerilen, kapsam dahilindeki mekanizmalar

Aşağıdaki maddeler ASVS için kapsam dahilindedir. Bunlar zorunlu hale getirilmemelidir ancak güvenli bir uygulamanın parçası olarak dikkate alınmaları şiddetle tavsiye edilir.

- Kullanıcıların daha güçlü bir parola belirlemesine yardımcı olmak için bir parola gücü ölçer sağlanmalıdır.
- Uygulamanın kök dizininde ya da .well-known dizininde, kullanıcıların güvenlik sorunları hakkında sahipleriyle iletişime geçebilecekleri bir bağlantı ya da e-posta adresini açıkça tanımlayan, herkese açık bir security.txt dosyası oluşturulmalıdır.
- İstemci tarafı girdi doğrulaması, güvenilir bir hizmet katmanında doğrulamaya ek olarak zorunlu kılınmalıdır. Çünkü bu, birisinin uygulamaya saldırmak amacıyla istemci tarafı kontrolleri atladığını keşfetmek için iyi bir fırsat sağlamaktadır.
- Bir robots.txt dosyası, X-Robots-Tag yanıt başlığı veya robots html meta etiketi kullanarak yanlışlıkla erişilebilen ve hassas sayfaların arama motorlarında görünmesi önlenmelidir.
- GraphQL kullanırken, yetkilendirmeyi her ayrı arayüzde ele almak zorunda kalmamak için yetkilendirme mantığı GraphQL veya çözümleyici katmanı yerine iş mantığı katmanında uygulanmalıdır.

Referanslar:

- RFC bağlantısı da dahil olmak üzere security.txt hakkında daha fazla bilgi

Yazılım Güvenliği İlkeleri

Aşağıdaki maddeler daha önce ASVS'de yer almaktaydı ancak tam olarak gereklilik değillerdir. Bunlar daha ziyade güvenlik kontrollerini uygularken göz önünde bulundurulması gereken ve takip edildiğinde daha sağlam kontrollere yol açacak ilkelerdir. Bunlar aşağıdakileri içerir:

- Güvenlik kontrolleri merkezi, basit (tasarım ekonomisi), doğrulanabilir şekilde güvenli ve yeniden kullanılabilir olmalıdır. Bu sayede mükerrer, eksik veya etkisiz kontrollerden kaçınılabilir.
- Mümkün olan her yerde, kontrolleri sıfırdan uygulamaya güvenmek yerine daha önce yazılmış ve iyi incelenmiş güvenlik kontrolü uygulamaları kullanılmalıdır.
- İdeal olarak, korunan verilere ve kaynaklara erişmek için tek bir erişim kontrol mekanizması kullanılmalıdır. Kopyala yapıştır veya güvensiz alternatif yollardan kaçınmak için tüm istekler bu tek mekanizmadan geçmelidir.
- Öznitelik veya özellik tabanlı erişim kontrolü, kodun kullanıcının sadece rolü yerine bir özellik veya veri öğesi için yetkisini kontrol ettiği önerilen bir modeldir. İzinler yine de roller kullanılarak tahsis edilmelidir.

Yazılım Güvenliği Süreçleri

ASVS 5.0'dan kaldırılan ancak hala iyi bir fikir olan bir dizi güvenlik süreci vardır. OWASP SAMM projesi bu süreçlerin nasıl etkili bir şekilde uygulanacağı konusunda iyi bir kaynak olabilir. Daha önce ASVS'de yer alan maddeler şunlardır:

- Geliştirmenin tüm aşamalarında güvenliği ele alan güvenli bir yazılım geliştirme yaşam döngüsünün kullanıldığı doğrulanmalıdır.
- Tehditleri belirlemek, karşı önlemleri planlamak, uygun risk yanıtlarını kolaylaştırmak ve güvenlik testlerine rehberlik etmek için her tasarım değişikliği veya sprint planlaması için tehdit modellemesinin kullanıldığı doğrulanmalıdır.
- Tüm kullanıcı hikayelerinin ve özelliklerinin işlevsel güvenlik kısıtlamaları içerdiği doğrulanmalıdır. "Bir kullanıcı olarak profilimi görüntüleyebilmeli ve düzenleyebilmeliyim. Başkasının profilini görüntüleyememeli veya düzenleyememeliyim."gibi kısıtlamalar örnek olabilir.
- Tüm geliştiriciler ve test uzmanları için güvenli kodlama kontrol listesi, güvenlik gereksinimleri, kılavuz veya politikanın mevcut olduğu doğrulanmalıdır.
- Uygulama kaynak kodunun arka kapılardan, kötü amaçlı kodlardan (örn. salami saldırıları, mantık bombaları, saatli bombalar) ve belgelenmemiş veya gizli özelliklerden (örn. Easter eggs, güvensiz hata ayıklama araçları) arındırılmış olmasını sağlamak için devam eden bir sürecin var olduğu doğrulanmalıdır. Bu bölüme uymak, üçüncü taraf kütüphaneleri de dahil olmak üzere kaynak koduna tam erişim olmadan mümkün değildir ve bu nedenle muhtemelen yalnızca en yüksek düzeyde güvenlik gerektiren uygulamalar için uygundur.
- Konuşlandırılmış ortamlarda yapılandırma kaymasını tespit etmek ve buna yanıt vermek için mekanizmaların mevcut olduğu doğrulanmalıdır. Bu, değişmez altyapı, güvenli bir taban çizgisinden otomatik yeniden dağıtım veya mevcut durumu onaylanmış yapılandırmalarla karşılaştıran sapma tespit araçlarının kullanılmasını içerebilir.
- Tüm üçüncü taraf ürünlerde, kütüphanelerde, çerçevelerde ve hizmetlerde kendi önerilerine göre yapılandırma sağlamlaştırması yapıldığı doğrulanmalıdır.

Referanslar:

- OWASP Threat Modeling Cheat Sheet
- OWASP Threat modeling
- OWASP Software Assurance Maturity Model Project
- Microsoft SDL

Ek E - Katkıda Bulunanlar

ASVS 4.0.0'ın yayınlanmasından bu yana yorum yapan veya pull request açan aşağıdaki kişilerin katkılarını minnetle kabul ediyoruz.

Herhangi bir hata fark ettiyseniz veya isminizin farklı görünmesini istiyorsanız, lütfen bize bildirin.

Johan Sydseter (sydseter)	luis servin (lfservin)	Oleksii Dovydkov (oleksiidov)	IZUKA Masahiro (maizuka)

James Sulinski (jsulinski)	Eli Saad (ThunderSon)	kkshitish9	Andrew van der Stock (vanderaj)
Rick M (kingthorin)	Bankde Eakasit (Bankde)	Michael Gargiullo (mgargiullo)	Raphael Dunant (Racater)
Cesar Kohl (cesarkohl)	inaz0	Joerg Bruenner (JoergBruenner)	David Deatherage (securitydave)
John Carroll (yosignals)	Jim Fenton (jimfenton)	Matteo Pace (M4tteoP)	Sebastien gioria (SPoint42)
Steven van der Baan (vdbaan)	Jeremy Bonghwan Choi (jeremychoi)	craig-shony	Riccardo Sirigu (ricsirigu)
Tomasz Wrobel (tw2as)	Alena Dubeshko (belalena)	Rafael Green (RafaelGreen1)	mjang-cobalt
clallier94	Kevin W. Wall (kwwall)	Jordan Sherman (jsherm-fwdsec / deleterepo)	Ingo Rauner (ingo-rauner)
Dirk Wetter (drwetter)	Moshe Zioni (moshe-apiiro)	Patrick Dwyer (coderpatros)	David Clarke (davidclarke-au)
Takaharu Ogasa (takaharuogasa)	Arkadii Yakovets (arkid15r)	Motoyasu Saburi (motoyasu-saburi)	leirn
wet-certitude	timhemel	RL Thornton (thornshadow99)	Thomas Bandt (aspnetde)
Roel Storms (roelstorms)	Jeroen Willemsen (commjoen)	anonymous-31	Kamran Saifullah (deFr0ggy)
Steve Springett (stevespringett)	Spyros (northdpole)	Hans Herrera (hansphp)	Marx314
CarlosAllendes	Yonah Russ (yruss972)	Sander Maijers (sanmai-NL)	Luboš Bretschneider (bretik)
Eva Sarafianou (esarafianou)	Ata Seren ataseren	Steve Thomas (Sc00bz)	Dominique RIGHETTO (righettod)
Steven van der Baan (svdb-ncc)	Michael Vacarella (Aif4thah)	Tonimir Kisasondi (tkisason)	Stefan Streichsbier (streichsbaer)
hi-unc1e	sb3k (starbuck3000)	mario-platt	Devdatta Akhawe (devd)
Michael Gissing (scolytus)	Jet Anderson (thatsjet)	Dave Wichers (davewichers)	Jonny Schnittger (JonnySchnittger)

Silvia Väli (silviavali)	jackgates73	1songb1rd	Timur - (timurozkul)
Gareth Heyes (hackvertor)	appills	suvikaartinen	chaals (chaals)
DanielPharos (AtlasHackert)	will Farrell (willfarrell)	Alina Vasiljeva (avasiljeva)	Paul McCann (ismisepaul)
Sage (SajjadPourali)	rbsec	Benedikt Bauer (mastacheata)	James Jardine (jamesjardine)
Mark Burnett (m8urnett)	dschwarz91	Cyber-AppSec (Cyber-AppSec)	Tib3rius
BitnessWise (bitnesswise)	damienbod (damienbod)	Jared Meit (jmeit-fwdsec)	Stefan Seelmann (sseelmann)
Brendan O'Connor (ussjoin)	Andrei Titov (andrettv)	Hans-Petter Fjeld (atluxity)	markehack
Neil Madden (NeilMadden)	Michael Geramb (mgeramb)	Osama Elnaggar (ossie-git)	mackowski
Ravi Balla (raviballa)	Hazana (hazanasec)	David Means (dmeans82)	Alexander Stein (tohch4)
BaeSenseii (baesenseii)	Vincent De Schutter (VincentDS)	S Bani (sbani)	Mitsuaki Akiyama (mak1yama)
Christopher Loessl (hashier)	victorxm	Michal Rada (michalradacz)	Veeresh Devireddy (drveresh)
MaknaSEO	darkzero2022	Liam (LiamDobbelaere)	Frank Denis (jedisct1)
Otto Sulin (ottosulin)	carllaw6885	Anders Johan Holmefjord (aholmis)	Richard Fritsch (rfricz)
mesutgungor	Scott Helme (ScottHelme)	Carlo Reggiani (carloreggiani)	Suyash Srivastava (suyash5053)
Mark Potter (markonweb)	Arjan Lamers (alamers)	Gøran Breivik (gobrtg)	flo-blg
Guillaume Déflache (guillaume-d)	Toufik Airane (toufik-airane)	Keith Hoodlet (securingdev)	Sinner (SoftwareSinner)
iloving	Jeroen Beckers (TheDauntless)	Joubin Jabbari (joubin)	yu fujioka (fujiokayu)
execjosh (execjosh)	Alicja Kario (tomato42)	Sidney Ribeiro (srjsoftware)	Gabriel Marquet (Gby56)

Drew Schulz (drschulz)	bedirhan	muralito	Ronnie Flathers (ropnop)
Philippe De Ryck (philippederyck)	Malte (mal33)	MazeOfThoughts	Andreas Falk (andifalk)
Javi (javixeneize)	Daniel Hahn (averell23)	borislav-c	Robin Wood (digininja)
miro2ns	Jan Dockx (jandockx)	vipinsaini434	priyanshukumar397
Nat Sakimura (sakimura)	Benjamin Häublein (BenjaminHae)	unknown-user-from	Ali Ramazan TAŞDELEN (alitasdln)
Pedro Escaleira (oEscal)	Josh (josh-hemphill)	Tim Würtele (SECtim)	AviD (avidouglen)
SheHacksPurple (shehackspurple)	fcerullo-cycubix	Hector Eryx Paredes Camacho (heryxpc)	Irene Michlin (irene221b)
Jonah Y-M (TG-Techie)	Dhiraj Bahroos (bahroos)	Jef Meijvis (jefmeijvis)	IzmaDoesItbeta
Abdessamad TEMMAR (TmmmmmR)	sectroyer	Soh Satoh (sohsatoh)	regoravalaz
james-t (james-bitherder)	Aram Hovsepyan (aramhovsepyan)	JaimeGomezGarciaSan	ValdiGit01
iwatachan (ishowta)	Vinod Anandan (VinodAnandan)	Kevin Kien (KevinKien)	paul-williamson-swoop
endergzr	Radhwan Alshamamri (Rado0z)	Grant Ongers (rewtd)	Cure53 (cure53)
AliR2Linux	Ads Dawson (Gang-GreenTemperTatum)	William Reyor (BillReyor)	gabe (gcrow)
mascotter	luissaiz	Suren Manukyan (vx-sec)	Piotr Gliźniewicz (pgliźniewicz)
Tadeusz Wachowski (tadeuszwachowski)	Nasir aka Nate (andesec)	settantasette	Lars Haulin (LarsH)
Terence Eden (edent)	JasmineScholz	Arun Sivadasan (teavanist)	Yusuf GÜR (yusuffgur)
Troy Marshall (troymarshall)	Tanner Prynn (tprynn)	Nick K. (nickific)	raoul361
Azeem Ilyas (TheAxZim)	Evo Stamatov (avioli)	Tim Potter (timpotter87)	Gavin Ray (GavinRay97)

monis (demideus)	Marcin Hoppe (MarcinHoppe)	Grambulf (ramshazar)	Jordan Pike (computersarebad)
Jason Rogers (jason-invision)	Ben Hall (benbhall)	JamesPoppyCock (jamesly123)	WhiteHackLabs (whitehacklabs)
Alex Gaynor (alex)	Filip van Laenen (filipvanlaenen)	jeurgen	GraoMelo
Andreas Kurtz (ay-kay)	Tom Tervoort (TomTervoort)	old man (deveras)	Marco Schnüriger (marcortw)
stiiin	infoseclearn (teaminfoseclearn)	hljupkij	Noe (nmarher)
Lyz (lyz-code)	Martin Riedel (mrtnrdl)	KIM Jaesuck (tcaesvk)	Barbara Schachner (bschach)
René Reuter (AresSec)	carhackpils	Tyler (tyler2cr)	Hugo (hasousa)
Wouter Bloeyaert (Someniak)	Mark de Rijk (markderijkinfosec)	Ramin (picohub)	Philip D. Turner (philipdturner)
Will Chatham (willc)			