# Web Uygulamalarında SQL Injection'a Karşı 2025 Yılında En Son ve En Etkili Güvenlik Teknikleri: Derinlemesine Bir Analiz

## Yönetici Özeti

SQL Injection (SQLi), 2025 yılında web uygulamaları için hâlâ ciddi ve "eskimeyen" bir tehdit olmaya devam etmektedir. Verizon DBIR'a göre, 2024 yılında veri ihlallerinin önemli bir kısmını web uygulaması saldırıları oluşturmuş ve bu saldırılar arasında SQLi'nin payı büyüktür. Özellikle bankacılık, finansal hizmetler ve sigorta (BFSI) gibi veri yoğun sektörlerde, SQLi saldırganlar için tercih edilen bir yöntem olarak öne çıkmaktadır.[1] Yirmi yılı aşkın süredir bilinen bir zafiyet olmasına rağmen, SQLi'nin kalıcılığı, eski uygulamalar, geliştirici hataları, gecikmiş yama süreçleri ve yapay zeka destekli saldırı araçlarının yükselişi gibi faktörlerle açıklanmaktadır.[1]

Bu rapor, SQL Injection'a karşı sağlam bir koruma sağlamak için temel kod düzeyindeki savunmaların, gelişmiş adaptif teknolojilerin ve Yazılım Geliştirme Yaşam Döngüsü (SDLC) boyunca stratejik entegrasyonun birleşiminin kritik olduğunu ortaya koymaktadır. Aşağıdaki tablo, 2025 yılı için en son ve en etkili 10 SQL Injection önleme tekniğini ve trendini özetlemektedir:

| Teknik Adı | Kısa Açıklama | Birincil Fayda | Ana Trend/Teknoloji | 2025 İçin Önemi |
|---|---|---|---|---|
| Parametreleştirilmiş Sorgular ve Hazırlanmış İfadeler | SQL kodunu veriden ayırarak, kullanıcı girdisinin yalnızca değer olarak işlenmesini sağlar. | SQL Injection'a karşı en güvenilir ve temel savunmayı sunar. | Temel Kod Güvenliği | Saldırıların temel nedenini ortadan kaldırır ve her zaman kritik öneme sahiptir. |
| Sağlam Girdi Doğrulama ve Sanitizasyon | Tüm kullanıcı girdilerinin beklenen formatlara ve değerlere uygunluğunu kontrol eder, kötü niyetli karakterleri temizler. | Kötü niyetli verilerin işlenmesini önler, API güvenliğinde anahtar rol oynar. | Veri Doğrulama, İzin Listesi Yaklaşımı | API tabanlı saldırı yüzeyinin büyümesiyle önemi artmaktadır. |

| | | | | |
|---|---|---|---|---|
| En Az Ayrıcalık Prensibi | Veritabanı hesaplarına ve uygulama kullanıcılarına yalnızca görevlerini yerine getirmek için gereken minimum erişim yetkisini verir. | Başarılı bir saldırının potansiyel etkisini sınırlar, hasarı azaltır. | Savunma Derinliği | Saldırılar daha sofistike hale geldikçe, ihlal durumunda hasarı en aza indirmek için hayati öneme sahiptir. |
| Saklı Prosedürlerin Güvenli Kullanımı | Veritabanında önceden derlenmiş SQL ifadeleridir; doğru şekilde parametreleştirildiğinde SQLi riskini azaltır. | SQL mantığını doğrudan kullanıcı etkileşiminden ayırır, performansı artırabilir. | Veritabanı Optimizasyonu ve Güvenliği | Parametreleştirme ile birlikte kullanıldığında veritabanı etkileşimini daha kontrollü hale getirir. |
| Yapay Zeka Destekli Web Uygulaması Güvenlik Duvarları (WAF'lar) ve Sanal Yama | Web uygulamaları ile internet arasında koruyucu bir kalkan görevi görür; yapay zeka ile gerçek zamanlı tehdit tespiti ve sanal yama sağlar. | Gelişen ve otomatikleşen SQLi saldırılarına karşı dinamik, gerçek zamanlı koruma sunar. | Yapay Zeka Destekli Güvenlik, Sanal Yama | Yapay zeka destekli saldırıların artmasıyla vazgeçilmez bir ön cephe savunmasıdır. |
| Tespit İçin Makine Öğrenimi ve Hibrit Yapay Zeka Modelleri | Kötü niyetli davranışları ve anormallikleri tespit etmek için makine öğrenimi ve hibrit yapay zeka modellerini kullanır. | Geleneksel imza tabanlı yöntemlerin yetersiz kaldığı sıfırıncı gün ve gelişen saldırıları tespit eder. | Makine Öğrenimi, Hibrit Yapay Zeka | Saldırıların karmaşıklığı ve otomasyonunun artmasıyla proaktif davranışsal analiz için kritik. |
| Güvenli Nesne-İlişkisel Eşleme (ORM) Uygulaması | ORM çerçevelerini doğru ve güvenli bir şekilde | ORM kullanımından kaynaklanan yanlış güvenlik | Geliştirici Sorumluluğu, Bağımlılık Güvenliği | Modern web geliştirmede ORM'lerin yaygınlığı |

| | | | | |
|---|---|---|---|---|
| | kullanma, parametreleştirme ve güncel tutma. | algısını ortadan kaldırır, soyutlama katmanındaki zafiyetleri önler. | | nedeniyle doğru kullanımı hayati önem taşır. |
| Kapsamlı API Güvenlik Önlemleri | API'ler için özel doğrulama, yetkilendirme, hız sınırlama ve API ağ geçitleri gibi çok katmanlı güvenlik önlemleri. | API'lerin artan saldırı yüzeyini korur, SQLi'nin yeni vektörlerini hedefler. | API Odaklı Güvenlik, Sıfır Güven | API trafiğinin web trafiğini geçmesiyle SQLi'nin yeni ve baskın hedefi haline gelmiştir. |
| Yazılım Geliştirme Yaşam Döngüsüne Güvenliği Entegre Etme (DevSecOps) | Güvenlik uygulamalarını ve araçlarını CI/CD süreçlerine entegre ederek, zafiyetleri geliştirmenin erken aşamalarında tespit eder. | Güvenlik açıklarını üretime ulaşmadan yakalar, düzeltme maliyetlerini azaltır. | Sol'a Kaydırma, Otomatik Güvenlik | Hızlı geliştirme döngülerinde proaktif güvenlik için vazgeçilmezdir. |
| Düzenli Güvenlik Denetimleri ve Dinamik Uygulama Güvenlik Testi (DAST) | Uygulamaları dışarıdan, bir saldırgan gibi test ederek gerçek dünya zafiyetlerini belirler; düzenli güvenlik denetimleri ve sızma testleri. | Uygulamaların gerçek saldırı yüzeyini ortaya koyar, SAST'ın eksiklerini tamamlar. | Saldırgan Görüşü Testi, Sürekli Güvenlik Testi | Karmaşık sistemlerde ve API'lerdeki sömürülebilir zafiyetleri tespit etmek için kritik. |

## 1. Giriş: 2025 Yılında SQL Injection'ın Gelişen Manzarası

SQL Injection (SQLi), web uygulamaları güvenliğinde yirmi yılı aşkın süredir bilinen bir zafiyet olmasına rağmen, 2025 yılında hâlâ en etkili ve yaygın sömürülen tekniklerden biri olarak varlığını sürdürmektedir.[1] Nitekim, OWASP Top 10 listesinde "Injection" (SQL, NoSQL, OS Komut vb. dahil) kritik bir zafiyet olarak yer almaktadır.[3] 2024 yılında, web

uygulaması saldırıları, tüm veri ihlallerinin %26'sını oluşturmuştur ve bu durum, SQLi'nin 2025'teki devam eden önemini ve tehlikesini açıkça göstermektedir.[1]

SQLi'nin bu denli uzun süredir etkili olmaya devam etmesi, sadece geliştiricilerin temel hatalarından kaynaklanan bir durum değildir; aynı zamanda daha geniş, sistemik sorunlara işaret etmektedir. Eski uygulamaların güncellenmesinin zorluğu, karmaşık sürekli entegrasyon/sürekli dağıtım (CI/CD) süreçlerinin güvenlik yamalarının zamanında uygulanmasını engellemesi ve yapay zeka destekli saldırı araçlarının ortaya çıkışı gibi faktörler, bu zafiyetin kalıcılığında önemli rol oynamaktadır.[1] Özellikle yapay zeka destekli saldırılar, SQLi sömürüsünü daha kolay ve yaygın hale getirmekte, hatta "Hizmet Olarak SQLi" gibi modellerle daha az yetenekli saldırganların bile sofistike saldırılar gerçekleştirmesine olanak tanımaktadır.[4] Bu durum, "eski, modası geçmiş anlamına gelmez" ifadesinin siber güvenlik bağlamında ne kadar doğru olduğunu göstermektedir.[1] SQLi'nin temel zafiyet konsepti aynı kalsa da, API'ler gibi yeni teknolojiler ve yapay zeka otomasyonu gibi yeni saldırı yöntemleri sayesinde saldırı yüzeyi sürekli olarak yenilenmekte veya genişlemektedir.

SQLi'nin devam eden yaygınlığına katkıda bulunan temel faktörler şunlardır:

- **Eski Uygulamalar:** Güncel olmayan platformlar genellikle zayıf kodlama uygulamalarına sahiptir ve güncellenmeleri zor olduğu için saldırılara açık kalmaktadır.[1]
- **Geliştirici Gözetimi:** Hızlı geliştirme döngüleri veya güvensiz kodun yeniden kullanılması, zafiyetlerin yeniden ortaya çıkmasına neden olabilmektedir.[1]
- **Gecikmiş Yama Uygulamaları:** Karmaşık CI/CD işlem hatları, güvenlik düzeltmelerinin zamanında uygulanmasını engelleyebilmektedir.[1]
- **Yapay Zeka Destekli Saldırılar:** Otomatik araçlar, SQLi zafiyetlerini büyük ölçekte taramak ve sömürmek için giderek daha fazla kullanılmaktadır, bu da saldırıları daha gizli ve otomatik hale getirmektedir.[1] Bu durum, daha az yetenekli kişilerin bile sofistike saldırılar başlatmasına olanak tanımaktadır.[4]

Bu rapor, web uygulamalarında SQL Injection'a karşı en etkili güvenlik tekniklerini ve 2025 yılı için öne çıkan trendleri derinlemesine incelemektedir. Seçim kriterleri, etkinlik, 2025 tehdit ortamına (yapay zeka destekli saldırılar ve API güvenliği dahil) uygunluk ve modern güvenli geliştirme uygulamaları içindeki benimsenme düzeyine öncelik vermektedir. Rapor, bu teknikleri temel kod düzeyindeki savunmalar, gelişmiş adaptif teknolojiler ve SDLC boyunca stratejik entegrasyon olmak üzere üç ana kategori altında ele almaktadır.

## 2. Temel Direkler: Temel Kod Düzeyinde Savunmalar

Bu teknikler, SQL Injection'ı önlemenin temelini oluşturur ve doğrudan enjeksiyon kusurlarının kök nedenini ele alan güvenli kodlama uygulamalarına odaklanır.

### 2.1. Parametreleştirilmiş Sorgular ve Hazırlanmış İfadeler

Bu teknik, SQL Injection'a karşı en güvenilir savunma yöntemi olarak kabul edilmektedir.[8] Hazırlanmış ifadeler, kabul edilebilir SQL kodunu önceden tanımlar ve ardından gelen sorgular için belirli parametreler belirler. Kullanıcı girdisi, kötü niyetli girdinin amaçlanan SQL mantığını değiştirmesini engelleyerek, yürütülebilir bir kod olarak değil, kesinlikle veri olarak işlenir.[3]

Bu yöntem, çeşitli programlama dilleri ve veritabanı sistemlerinde geniş çapta desteklenmektedir.[13] Örneğin, PHP'de PDO kullanımı, Hibernate'de adlandırılmış parametreler, SQLAlchemy'de bağlı parametrelerle text() işlevi, Django ORM'de QuerySet API'si, Entity Framework'te LINQ veya FromSqlInterpolated(), Sequelize'de replacements veya bind seçenekleri, Prisma'da $queryRaw etiketli şablon sözdizimi, Eloquent'te DB::select veya Model::where(), Active Record'da hash tarzı koşullar ve GORM'da Raw() sorgularında veya Where(...) içindeki yer tutucular bu yaklaşımın örnekleridir.[8]

2025 yılı için bu teknik, "altın standart" ve birincil savunma yöntemi olmaya devam etmektedir.[3] Etkinliği, kod ile veriyi temelden ayırmasından kaynaklanmakta olup, bu da onu sofistike enjeksiyon girişimlerine karşı bile dirençli kılmaktadır. Bu durum, yapay zeka ve WAF'lar gibi gelişmiş teknolojilerin savunma derinliği ve gelişen tehditleri yakalamak için kritik olmasına rağmen, temel güvenli kodlama uygulamalarının yeri doldurulamaz olduğunu göstermektedir. Bu uygulamalar, uygulama katmanının kendisinde saldırı yüzeyini azaltan ilk savunma hattını oluşturmaktadır. ORM'lerle bile geliştiricilerin parametreleştirmeyi açıkça sağlaması gerektiği vurgusu [8], bu temel prensibin vazgeçilmez olduğunu ortaya koymaktadır.

### 2.2. Sağlam Girdi Doğrulama ve Sanitizasyon

Bu teknik, verinin önceden belirlenmiş kriterlere göre incelenmesini ve biçimlendirilmesini (doğrulama) ve geçersiz veya güvensiz karakterlerin değiştirilmesini/kaldırılmasını (sanitizasyon) içerir.[10] Bu uygulama, tüm kullanıcı girdilerinin beklenen formatlara ve değerlere uygun olmasını sağlayarak kötü niyetli verilerin işlenmesini önler.[3]

En iyi uygulama, bilinen kötü niyetli kalıpları engellemeye çalışan bir "kara liste" (denylisting) yerine, yalnızca açıkça güvenilen değerlerin kabul edildiği bir "izin listesi" (whitelisting) yaklaşımını tercih etmektir.[10] Bu yaklaşım, özellikle tablo veya sütun adları

gibi parametreleştirilemeyen girdiler için etkilidir.[13] Temel kontroller arasında veri formatlarının doğrulanması, özel karakterlerin temizlenmesi, boyut sınırlarının uygulanması ve yürütülebilir kodun engellenmesi yer almaktadır.[14] Doğrulama, istek yaşam döngüsünün mümkün olan en erken aşamasında gerçekleşmelidir.[8]

2025 yılı için bu teknik, tüm kullanıcı tarafından sağlanan girdiler için vazgeçilmezdir, özellikle API trafiği büyüdükçe ve birincil saldırı vektörü haline geldikçe.[7] Parametreleştirme kullanıldığında bile, bu teknik kritik bir savunma katmanı sağlayarak diğer zafiyetlere yol açabilecek hatalı veya beklenmedik verileri yakalar.

### 2.3. En Az Ayrıcalık Prensibi

Bu prensip, veritabanı hesaplarına (ve uygulama kullanıcılarına) yalnızca belirli görevlerini yerine getirmek için gereken minimum erişim düzeyini vermeyi içerir.[9] Örneğin, bir uygulama yalnızca bir tablodan okumaya ihtiyaç duyuyorsa, veritabanı hesabının yazma, değiştirme veya silme izinleri olmamalıdır.[8]

Bu yaklaşım, bir SQLi saldırısı başarılı olursa bir saldırganın verebileceği potansiyel hasarı sınırlar ve tam veritabanı ele geçirilmesini veya yetkisiz veri değiştirme/silme işlemlerini önler.[8] Veritabanı yapılandırmasında, MSSQL'in ideal olarak minimum ayrıcalıklara sahip, özel, yönetici olmayan bir yerel hesap olarak çalıştırılması ve SQL Agent gibi SQL hizmetlerinin de kısıtlı izinlerle çalıştırılması önerilmektedir.[16]

2025 yılı için bu, kritik bir "savunma derinliği" stratejisidir.[13] Saldırılar daha sofistike ve otomatik hale geldikçe, başarılı bir ihlal her zaman bir olasılıktır. En az ayrıcalık, patlama yarıçapını en aza indirerek, ilk enjeksiyon başarılı olsa bile hassas verileri korur. Bu, teknik kontrolleri tamamlayan temel bir güvenlik prensibidir.

### 2.4. Saklı Prosedürlerin Güvenli Kullanımı

Saklı prosedürler, veritabanında depolanan önceden derlenmiş SQL ifadeleridir. Doğru uygulandığında, SQL mantığını doğrudan kullanıcı etkileşiminden ayırarak SQLi risklerini azaltabilirler.[9]

Ancak, kritik bir uyarı bulunmaktadır: Saklı prosedürlerin *kendileri* kullanıcı girdileri için parametreleştirilmiş ifadeler kullanmalıdır. Eğer dinamik olarak SQL dizeleri oluşturur ve girdiyi temizlemezlerse, diğer dinamik SQL gibi savunmasız hale gelirler.[11]

2025 yılı için, tek başına bir çözüm olmasa da, parametreleştirme ile birleştirildiğinde saklı prosedürler performans avantajları ve veritabanı üzerinde doğrudan gerçekleştirilebilecek işlem türlerini sınırlayarak belirli bir güvenlik düzeyi sunar.[10] Daha

yapılandırılmış ve kontrollü bir veritabanı etkileşim katmanına katkıda bulunurlar.

## 3. Gelişmiş ve Adaptif Savunmalar: Modern Güvenlik Teknolojilerinden Yararlanma

Bu teknikler, SQL Injection saldırılarının gelişen doğasına, özellikle yapay zeka destekli olanlara, karşı dinamik, gerçek zamanlı koruma ve akıllı sistemlere odaklanmaktadır.

### 3.1. Yapay Zeka Destekli Web Uygulaması Güvenlik Duvarları (WAF'lar) ve Sanal Yama

Yapay Zeka Destekli WAF'lar:
WAF'lar, web uygulamaları ve API'ler ile internet arasında koruyucu bir kalkan görevi görerek kötü niyetli HTTP/S trafiğini filtreler, izler ve engeller.7 Imperva WAF ve Cloudflare WAF gibi modern WAF'lar, gelişmiş güvenlik için yapay zeka ve makine öğrenimi teknolojilerinden yararlanmaktadır.10
Yapay zeka destekli WAF'lar, SQLi girişimlerini gerçek zamanlı olarak tespit edip engelleyebilir.[1] Davranışsal tehdit modellemesi kullanarak, daha gizli, otomatik saldırılar ve zaman gecikmeli tespit veya Python tabanlı komut dosyası enjeksiyonu gibi gelişmiş teknikler de dahil olmak üzere yeni ve gelişen SQLi yüklerine karşı savunmalarını uyarlarlar.[1] Örneğin, Cloudflare'ın WAF'ı, ağındaki trilyonlarca günlük istekten elde edilen tehdit istihbaratıyla sürekli olarak güncellenmektedir.[19] Nisan 2025'ten bir vaka çalışması, SiteWALL'ın yapay zekasının bir finans müşterisine yönelik 1.330 SQLi girişimini sıfır yanlış pozitif ve sıfır kesinti süresiyle başarıyla engellediğini göstermiştir.[1]

Saldırganlar yapay zekayı kullanarak yeni ve kaçamak SQLi yükleri oluşturdukça, savunmanın da yapay zeka destekli ve adaptif olması gerekmektedir. Yapay zeka destekli WAF'lar, statik imza tabanlı araçlardan, yapay zeka tarafından oluşturulan saldırıların karakteristik anomalili davranışlarını, belirli bir imza bilinmese bile tanımlayabilen dinamik, öğrenen sistemlere dönüşmektedir. Bu durum, onları 2025 tehdit ortamında vazgeçilmez bir ön cephe savunucusu konumuna getirmektedir.

Sanal Yama:
Sanal yama, ön saldırı zafiyet taraması yoluyla risk noktalarını belirlemeyi ve ardından uygulamanın kaynak kodunu değiştirmeden bu boşlukları gerçek zamanlı olarak kapatmayı içerir.1 Bu, kalıcı kod düzeltmeleri uygulanana kadar bir koruma katmanı sağlayarak, SQLi'nin gelişmesinin nedenlerinden biri olan gecikmiş yama sorununu hafifletir.1
WAF Güvenlik Modelleri:
WAF'lar tipik olarak üç güvenlik modeli kullanır:

| Model Adı | Temel Mekanizma | Avantajlar | Dezavantajlar | En İyi Kullanım Durumu |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Pozitif Güvenlik Modeli (İzin Listesi) | Varsayılan olarak tüm istekleri reddeder, yalnızca güvenilir olduğu bilinenleri (öğrenilmiş davranışlar) izin verir. Makine öğrenimi kullanır. | Bilinmeyen ve sıfırıncı gün saldırılarına karşı güçlü koruma. Daha az kaynak yoğun. | Öğrenme süresi gerektirir; yanlış ayarlanırsa yasal trafiği engelleyebilir. | İstikrarlı ve iyi tanımlanmış uygulama davranışlarına sahip sistemler. |
| Negatif Güvenlik Modeli (Engelleme Listesi) | Bilinen zafiyetlere karşı güncel imzaları kullanır; engellenecek trafik türlerini tanımlar, geri kalanını kabul eder. | Bilinen saldırılara karşı hızlı ve etkili koruma. | Sürekli güncelleme gerektirir; bilinmeyen (sıfırıncı gün) saldırılara karşı yetersiz kalabilir. | Bilinen ve yaygın saldırıların hedeflendiği genel uygulamalar. |
| Hibrit Yaklaşım | Hem izin hem de engelleme listelerinin güçlü yönlerini birleştirir; izin listelerini kullanırken yaygın saldırılar için ikincil bir engelleme listesi kontrolü ekler. | Hem bilinen hem de bilinmeyen potansiyel tehditlere karşı kapsamlı güvenlik sağlar. | Daha karmaşık yapılandırma ve yönetim gerektirebilir. | Yüksek güvenlik gerektiren, dinamik ve karmaşık web uygulamaları. |

2025 yılı için yapay zeka destekli WAF'lar ve sanal yama, özellikle SQLi saldırılarının artan karmaşıklığı ve otomasyonuna karşı dinamik, gerçek zamanlı tehdit tespiti ve yanıtı için kritik öneme sahiptir.[1] Kod düzeyindeki savunmaları atlayabilecek veya sıfırıncı gün zafiyetlerini sömürebilecek saldırıları yakalayarak önemli bir "savunma derinliği" katmanı sağlarlar.[13]

### 3.2. Tespit İçin Makine Öğrenimi ve Hibrit Yapay Zeka Modelleri

Makine öğrenimi (ML), siber güvenlikte güçlü bir araç olarak ortaya çıkmıştır ve yeni

veri kalıplarından ve anormalliklerinden öğrenme ve bunlara uyum sağlama yeteneği sunmaktadır.[4] ML tabanlı sistemler, geleneksel savunmaların gözden kaçırabileceği sofistike saldırılar da dahil olmak üzere, normdan sapan kötü niyetli davranışları tanımlayabilmektedir.[4]

2025 yılında yapılan araştırmalar (ICISSP 2025 bildirisi), SQLi tespiti ve hafifletilmesini geliştirmek için Naive Bayes, Uzun Kısa Süreli Bellek (LSTM) ve Random Forest gibi algoritmaları birleştiren yenilikçi hibrit ML modelleri önermektedir.[4] Diğer çalışmalar ise Destek Vektör Makineleri (SVM), Karar Ağaçları, Sinir Ağları ve Topluluk Öğrenimi modellerini araştırmaktadır.[20]

Bu hibrit yaklaşımlar, tespit doğruluğunda önemli iyileşmeler göstermektedir (örneğin, Naive Bayes + LSTM + Random Forest modeli için test setlerinde %99,89'luk bir doğruluk), bu da yanlış pozitifleri en aza indirmek için kritik öneme sahiptir.[4] Ayrıca, tüm SQL saldırı varyantları arasında genelleme yapma ve sıfırıncı gün SQLi saldırıları da dahil olmak üzere yeni, daha önce görülmemiş saldırı kalıplarına karşı yüksek performans sağlama yeteneğine odaklanmaktadırlar.[4] ML tabanlı tespit, aynı zamanda düşmanca sağlamlık (modelleri onları atlatmak için tasarlanmış saldırılara karşı dirençli hale getirme) ve açıklanabilir yapay zeka (XAI) gibi zorlukları da ele alarak yorumlanabilirliği artırmakta ve gerçek zamanlı tespit için ölçeklenebilirlik sağlamaktadır.[20]

Geleneksel imza tabanlı yöntemler, yeni saldırı kalıplarına uyum sağlamakta zorlanmaktadır.[4] Bu durum, tespitte temel bir paradigma kaymasını işaret etmektedir. Reaktif olarak bilinen kötü kalıpları (imzaları) tanımlamak yerine, ML modelleri "normal" davranışları ve sorgu yapılarını proaktif olarak öğrenmektedir. Bu, yeni veya sıfırıncı gün saldırıları için bile kötü niyetli niyeti gösteren sapmaları tanımlamalarına olanak tanır. Odak noktası, yapay zeka destekli saldırıların hızla yeni saldırı vektörleri oluşturabileceği durumlarda vazgeçilmez olan, sadece kalıp eşleştirmesi yerine davranışsal anomali tespitidir. Bu nedenle, ML ve hibrit yapay zeka modelleri, sık manuel kural güncellemelerine ihtiyaç duymadan gelişen tehdit ortamına otonom olarak uyum sağlayabilen dinamik, sağlam ve ölçeklenebilir bir çözüm sunmaktadır.[4] Bu, özellikle saldırıları daha az yetenekli kişilere erişilebilir kılan "Hizmet Olarak SQLi" tekliflerine karşı koymak için hayati öneme sahiptir.[4]

| **Algoritma/Model** | **Ana Güçlü Yönü** | **SQLi Tespitine Katkısı** | **Örnek (Varsa)** |
|---|---|---|---|
| Naive Bayes | Verimlilik, geniş veri kümelerini işleme | Kelime frekansına ve dağılımına dayalı | TF-IDF dönüştürülmüş veri |

| | | | |
|---|---|---|---|
| | yeteneği. | olarak kötü niyetli veya normal sınıf olasılıklarını hesaplar. | kümesi üzerinde MultinomialNB sınıflandırıcısı. |
| Uzun Kısa Süreli Bellek (LSTM) | Sıralı verileri işleme yeteneği, uzun vadeli bağımlılıkları öğrenme. | SQLi kalıplarını tanımak için girdi dizilerini (tokenleştirilmiş SQL sorguları) işler, son katmandan özellik temsilleri çıkarır. | Çift yönlü LSTM mimarisi, aşırı uyumu önlemek için dropout ve L2 düzenliliği ile geliştirilmiş. |
| Random Forest | Sınıflandırma görevlerinde doğruluk, aşırı uyuma karşı direnç. | Naive Bayes olasılıklarını ve LSTM'den türetilen özellikleri entegre ederek birleşik bir özellik kümesi oluşturur; karar ağaçları aracılığıyla en iyi özellik bölmelerini seçer. | Naive Bayes ve LSTM çıktılarının birleştirilmesiyle oluşturulan birleşik özellik kümesi üzerinde eğitilmiş topluluk modeli. |
| Destek Vektör Makineleri (SVM) | Yüksek boyutlu alanlarda etkili, net bir ayrım çizgisi olan durumlarda iyi performans. | Sorgu yapıları ve kullanıcı davranışlarındaki kalıpları kullanarak kötü niyetli sorguları sınıflandırır. | Denetimli öğrenme ile SQLi tespiti. |
| Sinir Ağları (Genel) | Karmaşık kalıpları ve korelasyonları öğrenme yeteneği. | Geniş veri kümeleri üzerinde eğitilerek, SQLi saldırılarının evrimleşen türlerine uyum sağlar. | Derin Evrişimsel Sinir Ağı (DCNN) tabanlı modeller. |
| Topluluk Öğrenimi Modelleri | Birden fazla algoritmanın birleştirilmesiyle genel performansı ve sağlamlığı artırma. | Farklı ML algoritmalarının güçlü yönlerini birleştirerek daha yüksek doğruluk ve genelleştirme yeteneği sağlar. | Naive Bayes, LSTM ve Random Forest'ın birleşimi. |

### 3.3. Güvenli Nesne-İlişkisel Eşleme (ORM) Uygulaması

Birçok geliştirici, ham SQL sorgularını soyutlayarak SQLi'ye karşı doğal bir koruma sağladığını varsayarak ORM çerçevelerine (örn. Hibernate, SQLAlchemy, Django ORM, Entity Framework, Sequelize, Prisma, Eloquent, Active Record, GORM) güvenmektedir.[5]

Ancak, ORM'ler mutlak bir çözüm değildir ve yanlış bir güvenlik hissi yaratabilir.[8] Zafiyetler hâlâ aşağıdaki yollarla ortaya çıkabilmektedir:

- **Ham SQL'in ORM İşlemleriyle Karıştırılması:** Geliştiricilerin karmaşık sorgular için ham SQL'e başvurması ve kullanıcı girdisinin bu ham SQL ifadelerinde düzgün bir şekilde işlenmemesi durumunda uygulama savunmasız hale gelir.[8]
- **ORM Özelliklerinin Yanlış Kullanımı:** Yerleşik parametre bağlama mekanizmaları yerine, sorguların manuel olarak dize birleştirme ile oluşturulması enjeksiyon kusurlarına yol açabilir.[8]
- **ORM Çerçevesinin Kendi İçindeki Zafiyetler:** ORM kütüphaneleri, diğer herhangi bir üçüncü taraf kodu gibi, kendi hatalarını veya CVE'lerini içerebilir (örneğin, Sequelize ve node-mysql'de SQLi zafiyetleri bulunmuştur).[8]
- **Güncel Olmayan ORM Sürümlerini Kullanma:** ORM bağımlılıklarını güncel tutmamak, uygulamaları zaten yamalanmış güvenlik açıklarına karşı savunmasız bırakır.[8]

Bu durum, güvenlik zafiyetlerinin soyutlama ile ortadan kalkmadığını, aksine genellikle yer değiştirdiğini göstermektedir. ORM'ler, sorgu oluşturmayı ele alarak doğrudan SQL enjeksiyon risklerini azaltırken, yanlış kullanıldığında yeni saldırı yüzeyleri (ORM enjeksiyonu) veya eski saldırı yüzeylerini yeniden ortaya çıkarabilirler. Sorumluluk, manuel SQL dize birleştirme hatalarından, ORM özelliklerinin doğru kullanımını sağlamaya ve ORM kütüphanesinin kendisini güvenli ve güncel tutmaya kaymaktadır. Bu, geliştiricilerin, soyutlamaya körü körüne güvenmek yerine, seçtikleri ORM'in SQL'i nasıl oluşturduğunu ve girdiyi nasıl işlediğini daha derinlemesine anlamaları gerektiği anlamına gelmektedir.

**ORM Kullanımında SQLi Önleme İçin En İyi Uygulamalar:**

- **Daima Parametreleştirilmiş Sorgular Kullanın:** ORM'lerle bile bu en güvenilir savunmadır. Parametre bağlama için yerleşik mekanizmaları kullanın.[8]
- **ORM'in Yerleşik Güvenlik Özelliklerini Kullanın:** Sorgu oluşturucuları, model tabanlı erişim desenlerini ve parametre bağlamayı otomatik olarak ele alan akıcı API'leri kullanın.[8]
- **Girdiyi Doğrulayın ve Temizleyin:** Kullanıcı girdilerinin beklenen formatlara

uygun olduğundan emin olarak ek bir savunma katmanı ekleyin.[8]

- **ORM Çerçevelerini Güncel Tutun:** En son hata düzeltmelerinden ve güvenlik yamalarından yararlanmak için ORM kütüphanelerini düzenli olarak güncelleyin.[8] npm audit, pip-audit veya Snyk gibi Yazılım Bileşimi Analizi (SCA) araçlarını kullanın.[8]
- **Uygun Erişim Kontrolleri Uygulayın:** Bir enjeksiyon meydana gelirse hasarı sınırlamak için veritabanı kullanıcı izinlerini sınırlayın.[8]

| Hata | Açıklama | Güvenli Uygulama | Örnek (Kod/Konsept) |
|---|---|---|---|
| Ham SQL'in ORM İşlemleriyle Karıştırılması | Geliştiricilerin karmaşık sorgular için doğrudan SQL kullanması ve kullanıcı girdisini bu ham SQL'de düzgün işlememesi. | Ham SQL kullanılması gerektiğinde bile, daima ORM'in parametreleştirme özelliklerini (örn. SQLAlchemy'de text() ile bağlı parametreler) kullanın, asla dize birleştirme yapmayın. | Ham SQL'de kullanıcı girdisini doğrudan birleştirmek yerine: SELECT * FROM users WHERE name = :name ve parametreyi bağlamak. |
| ORM Özelliklerinin Yanlış Kullanımı (Parametreleştirmeyi Atlamak) | ORM'in yerleşik parametre bağlama mekanizmaları yerine, sorguları manuel olarak dize birleştirme ile oluşturmak. | ORM'in yerleşik güvenlik özelliklerini (sorgu oluşturucular, model tabanlı erişim desenleri) kullanarak otomatik parametre bağlamayı sağlayın. | User::where('email', $userInput)->first(); (Eloquent) veya .filter(email=user_input) (Django) gibi ORM'in güvenli API'lerini kullanmak. |
| ORM Çerçevesinin Kendi İçindeki Zafiyetler | ORM kütüphanelerinin (örn. Sequelize, node-mysql) kendi içinde güvenlik açıkları içermesi. | ORM kütüphanelerini ve bağımlılıklarını düzenli olarak güncel tutun. npm audit, pip-audit veya SCA araçları (örn. Snyk) kullanarak bilinen zafiyetleri denetleyin. | npm audit komutunu düzenli çalıştırmak ve tespit edilen zafiyetleri gidermek. |
| Güncel Olmayan ORM Sürümlerini Kullanma | ORM bağımlılıklarını güncel tutmamak, zaten yamalanmış güvenlik açıklarına karşı uygulamaları | Güvenlik yamalarından ve hata düzeltmelerinden yararlanmak için ORM kütüphanelerinizi | Projenizin bağımlılıklarını düzenli olarak kontrol etmek ve en son güvenli sürümlere |

| | | | |
|---|---|---|---|
| | savunmasız bırakır. | sürekli güncelleyin. | yükseltmek. |
| Yetersiz Erişim Kontrolleri | Uygulamanın veritabanına, ihtiyaç duyduğundan daha fazla yetkiyle erişmesi. | Veritabanı kullanıcılarına yalnızca gerekli minimum izinleri verin (DROP, ALTER gibi hassas işlemlere erişim olmamalıdır). | Bir uygulamanın sadece okuma iznine ihtiyacı varsa, veritabanı kullanıcısına sadece okuma yetkisi vermek. |

### 3.4. Kapsamlı API Güvenlik Önlemleri

2025 yılında API trafiği, geleneksel web trafiğini geride bırakarak siber saldırılar için en sık giriş noktası haline gelmiştir.[7] "API yayılımı" olarak bilinen yönetilmeyen API'ler, sömürü için birincil hedefler oluşturmaktadır.[15]

API'ler, girdi verileri doğrulanmaz veya temizlenmezse SQL, NoSQL ve OS komut enjeksiyonları gibi klasik enjeksiyon saldırılarına karşı hâlâ savunmasızdır.[7] Saldırganlar, geleneksel SQLi'ye benzer şekilde, API istekleri aracılığıyla bozuk kod enjekte etmektedir.[7]

2025'teki temel API güvenlik trendleri şunlardır:

- **API'ler İçin Sıfır Güven:** "Asla güvenme, daima doğrula" modelini API ekosistemlerine genişletmek, sürekli doğrulama ve en az ayrıcalıklı erişimi zorunlu kılmak.[15]
- **Yapay Zeka Destekli Tehdit Tespiti:** Güvenlik araçları, enjeksiyon girişimleri gibi tehditleri daha hızlı tanımlamak için API davranışındaki anormallikleri gerçek zamanlı olarak tespit etmek için yapay zeka/makine öğrenimi kullanmaktadır.[15]
- **Kod Olarak API Güvenliği:** Güvenliği DevOps işlem hatlarına yerleştirmek, OpenAPI gibi araçları kullanarak politikaları ve test senaryolarını erken tanımlamak.[15]
- **Birleşik API Yönetim Platformları:** API yayılımını ve uyumluluğu ele almak için API keşfi, testi, izlemesi ve tehdit tespiti gibi çeşitli API güvenlik işlevlerini birleştirmek.[15]

API'lerin birincil saldırı giriş noktası haline gelmesi, SQLi'nin eski bir zafiyet olmasına rağmen, API odaklı dünyada yeni bir yaşam ve saldırı vektörleri bulduğunu göstermektedir. Artık sadece geleneksel web formlarını güvence altına almakla kalmayıp, her API uç noktasını güvence altına almak gerekmektedir. Bu durum, genel web uygulaması savunmalarının ötesine geçen özel API güvenlik önlemlerini, örneğin

API'ye özgü girdi doğrulamasını, API protokollerini anlayan API ağ geçitlerini ve API'ye duyarlı WAF'ları zorunlu kılmaktadır. API etkileşimlerinin ölçeği ve otomasyonu, yapay zeka destekli tespiti bu alanda özellikle kritik hale getirmektedir.

SQLi ile ilgili modern API savunma mekanizmaları şunları içerir:

- **Girdi Doğrulama ve Sanitizasyon:** Tüm girdileri beklenen formatlara göre doğrulayarak ve temizleyerek enjeksiyon saldırılarını önlemek için kritik öneme sahiptir.[7] Güvenli değerleri beyaz listelemeyi tercih etmek önemlidir.[15]
- **API Ağ Geçitleri ve WAF'lar:** API ağ geçitleri, trafiğin merkezi giriş noktası olarak hareket eder, kimlik doğrulama, izleme ve trafik yönetimini ele alır.[7] API'ye özgü kurallara sahip WAF'lar, SQLi girişimleri de dahil olmak üzere kötü niyetli istekleri filtreler.[7]
- **Hız Sınırlama ve Kısıtlama:** Hizmet reddi ve API kötüye kullanımını önler, belirli süreler boyunca istek sayısını sınırlar.[7]
- **Kimlik Doğrulama ve Yetkilendirme:** OAuth 2.0, OpenID Connect, token tabanlı kimlik doğrulama ve ayrıntılı kontrol için RBAC/ABAC gibi güçlü mekanizmalar.[7]

2025 yılı için API odaklı mimarilere geçiş, API'leri güvence altına almayı SQLi önlemesi için ikincil bir düşünce olmaktan çıkarıp birincil bir endişe haline getirmektedir. Özel girdi doğrulama ve API'ye duyarlı WAF'lar da dahil olmak üzere kapsamlı API güvenliği, bu genişleyen saldırı yüzeyini korumak için vazgeçilmezdir.[7]

## 4. Stratejik Entegrasyon: SDLC Boyunca Proaktif Güvenlik

Bu teknikler, güvenliği yazılım geliştirme yaşam döngüsünün başından ve sürekli olarak entegre etmeyi vurgulamaktadır.

### 4.1. Yazılım Geliştirme Yaşam Döngüsüne Güvenliği Entegre Etme (DevSecOps)

DevSecOps, güvenlik uygulamalarını ve araçlarını CI/CD işlem hattının her aşamasına yerleştirerek geliştirme sürecinde güvenliği "sola kaydırma" konseptini ifade eder.[3] Bu proaktif yaklaşım, geliştirme sırasında ortaya çıkan güvenlik sorunlarının sayısını azaltır.[23] Uygulamalar arasında, CI/CD işlem hatlarında DevSecOps prensiplerinin uygulanması [3] ve güvenlik kontrollerinin ve test araçlarının erken ve sürekli entegrasyonu yer almaktadır.[15] "Kod Olarak API Güvenliği", OpenAPI gibi araçlar kullanılarak güvenlik politikalarının ve test senaryolarının erken tanımlandığı önemli bir trenddir.[15]

Bu yaklaşım, reaktif "bul-ve-düzelt" modelinden (zafiyetlerin geç keşfedildiği ve yamalamanın geciktiği) proaktif "erken önle" modeline stratejik bir geçişi temsil etmektedir. Güvenliği otomatikleştirilmiş işlem hatlarına entegre ederek, kuruluşlar

SQLi önlemesi de dahil olmak üzere güvenli kodlama uygulamalarının büyük ölçekte ve sürekli olarak uygulanmasını sağlayabilirler. Bu, manuel kontroller veya dağıtım sonrası denetimlere güvenmek yerine, geliştirici gözetimi ve gecikmiş yama uygulama gibi SQLi'nin kalıcılığına katkıda bulunan faktörleri doğrudan ele almaktadır.

2025 yılı için, modern geliştirme ve dağıtımın hızlı temposu göz önüne alındığında, güvenliği baştan itibaren entegre etmek büyük önem taşımaktadır. Bu, SQLi zafiyetlerini üretime ulaşmadan yakalamaya yardımcı olur, düzeltme maliyetlerini ve riskleri azaltır.[23] Güvenliğin sadece güvenlik ekibinin değil, herkesin ortak sorumluluğu olduğu bir kültürü teşvik eder.

### 4.2. Düzenli Güvenlik Denetimleri ve Dinamik Uygulama Güvenlik Testi (DAST)

Düzenli Güvenlik Denetimleri ve Sızma Testleri:
Bu mekanizmalar, zafiyetleri belirlemek ve yamalamak için sürekli değerlendirmeler ve manuel incelemeler içerir.7 Sızma testi, karmaşık, zincirleme zafiyetleri ortaya çıkarmak için saldırgan eylemlerini simüle eden insan test uzmanlarını içerir.23 2025 yılı için, sağlam bir güvenlik duruşunu sürdürmek ve potansiyel zafiyetleri sömürülmeden önce belirlemek için vazgeçilmezdir.12 Otomatik araçları tamamlayarak mantık hatalarını veya sofistike atlatmaları bulmaya yardımcı olur.
Dinamik Uygulama Güvenlik Testi (DAST):
DAST, çalışan uygulamaları dışarıdan içeriye doğru analiz ederek, SQLi, XSS, güvensiz yapılandırmalar ve kimlik doğrulama sorunları gibi zafiyetleri belirlemek için kötü niyetli bir saldırganı simüle eder.23 Uygulama girdilerini ve yanıtlarını inceleyerek sömürülebilir zafiyetleri bulur.24
DAST'ın avantajları arasında, çalışan uygulamadaki zafiyetleri doğruladığı için Statik Uygulama Güvenlik Testi (SAST)'ne göre daha az yanlış pozitif üretmesi yer almaktadır.[23] Saldırganların gerçekten ne gördüğüne ve sömürebileceğine odaklanır.[24] Bu, karmaşık bir ekosistemde "saldırganın bakış açısıyla" test yapmanın kritik gerekliliğini vurgulamaktadır. SAST, kod hatalarını bulmak için önemli olsa da, DAST bu hataların çalışan bir uygulamada gerçekten sömürülebilir olup olmadığını doğrular, böylece gürültüyü azaltır ve düzeltme çabalarını gerçek tehditlere odaklar. Birden fazla bileşenin etkileşimde bulunduğu ve yanlış yapılandırmaların zafiyetlere yol açabileceği karmaşık web uygulaması ve API ekosistemlerinde, DAST, SQLi için gerçek dünya saldırı yüzeyinin en doğru resmini sunar.

DAST araçları, sık güncellenen uygulamalar için CI/CD işlem hatlarına entegre edilebilir.[23] Kapsamlı kapsama için DAST'ı SAST, Yazılım Bileşimi Analizi (SCA) ve manuel sızma testi gibi diğer test yöntemleriyle birleştirmek önerilmektedir.[23] 2025 yılı için popüler DAST araçları arasında OWASP ZAP, Burp Suite Professional, Acunetix, Netsparker, Invicti, Checkmarx DAST, Tenable WAS, Qualys WAS, Black Duck DAST ve

Veracode Dynamic Analysis yer almaktadır.[3]

2025 yılı için DAST, canlı uygulamalardaki uzaktan sömürülebilir SQLi zafiyetlerini belirlemek ve gerçekçi bir güvenlik duruşu görünümü sağlamak için kritik öneme sahiptir.[24] DevOps iş akışlarına entegrasyonu, SDLC boyunca sürekli güvenlik testini sağlar.

## 5. Gelecek Görünümü ve Stratejik Öneriler

SQL Injection saldırıları ve savunma mekanizmalarında 2025 sonrası için beklenen trendler şunlardır:

- **Yapay Zeka Destekli Saldırıların Artan Karmaşıklığı:** Saldırganlar, daha polimorfik ve kaçamak SQLi yükleri oluşturmak için üretken yapay zekadan yararlanmaya devam edecek, bu da imza tabanlı tespitin etkinliğini daha da azaltacaktır.[1] "Hizmet Olarak SQLi" modelinin yaygınlaşması, saldırganlar için giriş engelini düşürecektir.[4]
- **API'ye Özgü SQLi'ye Odaklanma:** API'ler baskın iletişim arayüzü haline geldikçe, saldırganlar enjeksiyon için giderek daha fazla API uç noktalarını hedef alacak, bu da daha özel API güvenlik çözümleri gerektirecektir.[7]
- **ML/Yapay Zeka Savunmalarının Evrimi:** Hibrit ML modelleri, gerçek zamanlı yeteneklere, düşmanca sağlamlığa ve açıklanabilir yapay zekaya odaklanarak gelişen tehditlere karşı koymak için daha da geliştirilecektir.[4]
- **Sıfır Güvenin Daha Geniş Benimsenmesi:** Sıfır Güven modeli, uygulama ve API güvenlik mimarilerine daha derinlemesine entegre edilecek, sürekli doğrulama ve en az ayrıcalık prensibini zorunlu kılacaktır.[15]
- **Otomatik Sanal Yama ve Kendi Kendini İyileştiren Sistemler:** Gerçek zamanlı, otomatik savunmaya yönelik eğilim, daha sofistike sanal yama çözümlerine ve potansiyel olarak zafiyetleri dinamik olarak hafifletebilen kendi kendini iyileştiren uygulamalara yol açacaktır.[1]

Kuruluşların dayanıklı web uygulamaları oluşturmaları için bütünsel öneriler şunlardır:

- **Temel Güvenliğe Öncelik Verin:** Parametreleştirilmiş sorguları, sağlam girdi doğrulamasını (izin listesi) ve en az ayrıcalık prensibini vazgeçilmez kodlama standartları olarak sürekli olarak uygulayın. Bunlar, en etkili ilk savunma hattı olmaya devam etmektedir.[3]
- **Adaptif Teknolojilere Yatırım Yapın:** Gelişen ve sıfırıncı gün SQLi saldırılarına karşı gerçek zamanlı, adaptif koruma sağlamak için yapay zeka destekli WAF'lar ve güçlü güvenlik yeteneklerine sahip API Ağ Geçitleri dağıtın.[1]
- **DevSecOps'u Benimseyin:** SDLC'nin her aşamasına güvenliği entegre edin.

Zafiyetleri erken ve sürekli yakalamak için CI/CD işlem hatlarında güvenlik testlerini (SAST, DAST, SCA) otomatikleştirin.[3]

- **Güvenli ORM Uygulaması:** Geliştiricileri ORM'in yanlış kullanım riskleri konusunda eğitin ve açık parametreleştirme ve ORM kütüphanelerini güncel tutma gibi en iyi uygulamaları zorunlu kılın.[8]
- **Düzenli Denetimler ve Sızma Testleri:** Karmaşık, zincirleme zafiyetleri belirlemek ve sağlam bir güvenlik duruşu sağlamak için otomatik araçları düzenli manuel güvenlik denetimleri ve sızma testleriyle tamamlayın.[7]
- **Güncel Kalın ve Eğitim Verin:** En son CVE'leri ve güvenlik araştırmalarını sürekli olarak izleyin. Geliştirme ve güvenlik ekiplerine güvenli kodlama uygulamaları ve gelişen tehditler hakkında sürekli eğitim sağlayın.[3]
- **Uyumluluğa Uyum:** Tüm güvenlik önlemlerinin, SQLi'ye karşı koruma gerektiren GDPR ve PCI DSS gibi ilgili uyumluluk düzenlemeleriyle uyumlu olduğundan emin olun.[3]

## 6. Sonuç

SQL Injection, yaşına rağmen, eski sistemler, geliştirici zorlukları ve yapay zeka destekli saldırıların yükselişiyle beslenen, 2025 yılında kritik ve gelişen bir tehdit olmaya devam etmektedir. Etkili savunma, temel güvenli kodlama uygulamalarını, gelişmiş yapay zeka destekli güvenlik teknolojilerini ve tüm yazılım geliştirme yaşam döngüsü boyunca proaktif entegrasyonu birleştiren çok katmanlı, bütünsel bir strateji gerektirmektedir. Kuruluşların, hassas verileri korumak, uygulama kullanılabilirliğini sürdürmek ve kalıcı ve sofistike SQLi tehditleri karşısında kullanıcı güvenini sağlamak için güvenlik duruşlarını sürekli olarak uyarlamaları gerekmektedir.

**Alıntılanan çalışmalar**


1. SQL Injection: An Evergreen Threat with Real-World Consequences ..., erişim tarihi Mayıs 31, 2025, https://www.sitewall.net/sql-injection-in-2025-ai-waf-defense/
2. The Art of Exploiting SQL Injection - Black Hat, erişim tarihi Mayıs 31, 2025, https://www.blackhat.com/html/bh-us-12/training/courses/bh-us-12-training_exploiting-sql-injection.html
3. Understanding OWASP Top 10 Vulnerabilities in 2025 with Real ..., erişim tarihi Mayıs 31, 2025, https://www.webasha.com/blog/understanding-owasp-top-10-vulnerabilities-with-real-world-examples-and-prevention-tips
4. www.scitepress.org, erişim tarihi Mayıs 31, 2025, https://www.scitepress.org/Papers/2025/130781/130781.pdf
5. The Detection and Defense Mechanism for SQL Injection Attack Based on Web Application, erişim tarihi Mayıs 31, 2025, https://www.researchgate.net/publication/362457643_The_Detection_and_Defens

e_Mechanism_for_SQL_Injection_Attack_Based_on_Web_Application
6. From RSA to Black Hat and DEF CON: Navigating the Top Cybersecurity Conferences of 2023 | Walker Sands, erişim tarihi Mayıs 31, 2025, https://www.walkersands.com/about/blog/from-rsa-to-black-hat-and-def-con-navigating-the-top-cybersecurity-conferences-of-2023/
7. The Rising Threat of API Attacks: How to Secure Your APIs in 2025 - Hackread, erişim tarihi Mayıs 31, 2025, https://hackread.com/rising-threat-of-api-attacks-how-to-secure-apis-2025/
8. SQL Injection in the Age of ORM: Risks, Mitigations, and Best ..., erişim tarihi Mayıs 31, 2025, https://afine.com/sql-injection-in-the-age-of-orm-risks-mitigations-and-best-practices/
9. What Is SQL Injection And How To Prevent Attacks | 2025 - Cyble, erişim tarihi Mayıs 31, 2025, https://cyble.com/knowledge-hub/what-is-sql-injection/
10. How to prevent SQL injection | Cloudflare, erişim tarihi Mayıs 31, 2025, https://www.cloudflare.com/learning/security/threats/how-to-prevent-sql-injection/
11. How to prevent SQL injection attacks with secure programming? - Tencent Cloud, erişim tarihi Mayıs 31, 2025, https://www.tencentcloud.com/techpedia/103611
12. SQL Injection in 2025: How One Vulnerability Can Expose ... - Cyserch, erişim tarihi Mayıs 31, 2025, https://www.cyserch.com/blog/SQL-Injection-in-2024:-How-One-Vulnerability-Can-Expose-Your-Business-Data
13. SQL Injection Prevention: 6 Strategies - Legit Security, erişim tarihi Mayıs 31, 2025, https://www.legitsecurity.com/aspm-knowledge-base/how-to-prevent-sql-injection
14. API Security Best Practices for Enterprise Protection in 2025 - Netguru, erişim tarihi Mayıs 31, 2025, https://www.netguru.com/blog/api-security
15. The State of API Security in 2025: Emerging Threats and Best ..., erişim tarihi Mayıs 31, 2025, https://securemyorg.com/2025/04/05/the-state-of-api-security-in-2025/
16. 11 Steps to Secure SQL in 2025 - UpGuard, erişim tarihi Mayıs 31, 2025, https://www.upguard.com/blog/11-steps-to-secure-sql
17. Claranet's top 10 web application vulnerabilities found in 2024, erişim tarihi Mayıs 31, 2025, https://www.claranet.com/us/blog/2025-01-07-claranet%E2%80%99s-top-10-web-application-vulnerabilities-found-2024
18. What Is A WAF? 2025 Guide to Web Application Firewalls | Radware, erişim tarihi Mayıs 31, 2025, https://www.radware.com/cyberpedia/application-security/what-is-waf/
19. Best 12 Web Application Firewall Software In 2025 - IO River, erişim tarihi Mayıs 31, 2025, https://www.ioriver.io/blog/best-web-application-firewall-software
20. (PDF) A STUDY OF MACHINE LEARNING-BASED APPROACHES FOR SQL INJECTION DETECTION AND PREVENTION - ResearchGate, erişim tarihi Mayıs 31, 2025,

https://www.researchgate.net/publication/389772600_A_STUDY_OF_MACHINE_LEARNING-BASED_APPROACHES_FOR_SQL_INJECTION_DETECTION_AND_PREVENTION
21. (PDF) Enhancing SQL Injection Attack Prevention: A Framework for Detection, Secure Development, and Intelligent Techniques - ResearchGate, erişim tarihi Mayıs 31, 2025, https://www.researchgate.net/publication/385697949_Enhancing_SQL_Injection_Attack_Prevention_A_Framework_for_Detection_Secure_Development_and_Intelligent_Techniques
22. Testing for ORM Injection - WSTG - Latest | OWASP Foundation, erişim tarihi Mayıs 31, 2025, https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05.7-Testing_for_ORM_Injection
23. Dynamic application security testing (DAST): The Ultimate Guide 2025 - CyberNX, erişim tarihi Mayıs 31, 2025, https://www.cybernx.com/dynamic-application-security-testing-guide/
24. Top 10 Dynamic Application Security Testing (DAST) Tools for 2025 - Invicti, erişim tarihi Mayıs 31, 2025, https://www.invicti.com/blog/web-security/10-best-dast-tools/