# Testing for NoSQL Injection (NoSQL Enjeksiyonu için Test)

## Summary (Özet)

NoSQL veritabanları, geleneksel SQL veritabanlarından daha gevşek tutarlılık kısıtlamaları sağlar. Daha az ilişkisel kısıtlamalar ve tutarlılık kontrolü gerektirerek, NoSQL veritabanları genellikle
performans ve ölçeklendirme avantajları sunar. Yine de bu veritabanları, geleneksel SQL sözdizimi kullanmasalar bile, enjeksiyon saldırılarına karşı potansiyel olarak savunmasızdır. Bu NoSQL enjeksiyon saldırıları, deklaratif SQL dilinden ziyade prosedürel bir dil içinde yürütülebileceğinden, potansiyel etkiler geleneksel SQL enjeksiyonundan daha büyüktür.

NoSQL veritabanı aramaları, uygulamanın programlama dilinde, özel bir API çağrısına yazılır veya ortak bir sözleşmeye göre biçimlendirilir (örneğin `XML`, `JSON`, `LINQ`, vb. , vb. Bu spesifikasyonları hedefleyen kötü amaçlı girdi, öncelikle uygulama sanitasyon kontrollerini tetiklemeyebilir. Örneğin, ortak HTML özel karakterleri filtrelemek gibi `< > &;`Özel karakterlerin dahil olduğu bir JSON API'sine karşı saldırıları engellemeyecek `/{}:`. . Şu anda bir uygulama içinde kullanılmak üzere 150'den fazla NoSQL veritabanı mevcuttur ve çeşitli dillerde ve ilişki modellerinde API'ler sağlar. Her biri farklı özellikler ve kısıtlamalar sunar. Aralarında ortak bir dil olmadığından, örneğin enjeksiyon kodu tüm NoSQL veritabanlarında uygulanmaz. Bu nedenle, NoSQL enjeksiyon saldırıları için test yapan herkesin belirli testler yapmak için kendilerini syntak, veri modeli ve altta yatan programlama dili ile tanıştırması gerekecektir.

NoSQL enjeksiyon saldırıları, bir uygulamanın geleneksel SQL enjeksiyonundan farklı alanlarda gerçekleştirilebilir. SQL enjeksiyonu veritabanı motorunda yürütüleceği durumlarda, NoSQL varyantları, NoSQL API kullanıcısı ve veri modeline bağlı olarak uygulama katmanı veya veritabanı katmanı sırasında yürütebilir. Tipik olarak NoSQL enjeksiyon saldırıları, saldırı dizisinin bir NoSQL API çağrısına ayrıştırıldığı, değerlendirildiği veya bulunduğu yerlerde gerçekleştirilir.

Ek zamanlama saldırıları, bir NoSQL veritabanındaki eşzamanlı kontrollerin olmamasıyla ilgili olabilir. Bunlar enjeksiyon testi kapsamında değildir.

MongoDB'nin yazıldığı sırada en çok kullanılan
NoSQL veritabanıdır ve bu nedenle tüm örnekler MongoDB API'leri içerecektir.

## How to Test (Nasıl Test Edilir)

## Testing for NoSQL Injection Vulnerabilities in MongoDB (MongoDB'de NoSQL Enjeksiyonu Güvenlik Açıkları için Test)

MongoDB API, BSON (İkili JSON) aramaları bekliyor ve güvenli bir BSON sorgu montaj aracı içeriyor. Bununla birlikte, MongoDB belgelerine göre - seri dışı JSON ve JavaScript ifadelerine birkaç alternatif sorgu parametresinde izin verilir. En çok kullanılan API çağrısı, keyfi JavaScript girişi sağlayandır. `$where` Operatör.

The MongoDB Altyazıları `$where` Operatör genellikle SQL içinde olduğu gibi basit bir filtre veya kontrol olarak kullanılır.

```
db.myCollection.find( { $where: "this.credits == this.debits" } );
```

İsteğe bağlı olarak JavaScript daha ileri koşullara izin vermek için de değerlendirilir.

```
db.myCollection.find( { $where: function() { return obj.credits - obj.debits < 0; } } );
```

### Example 1 (Örnek 1)

Bir saldırgan içeri giren verileri manipüle edebildiyse `$where` Operatör, bu saldırgan MongoDB sorgusunun bir parçası olarak değerlendirilecek keyfi JavaScript içerebilir. Bir örnek, aşağıdaki kodda, kullanıcı girişi doğrudan sanitizasyon olmadan MongoDB sorgusuna aktarılırsa, güvenlik açığı ortaya çıkar.

```
db.myCollection.find( { active: true, $where: function() { return obj.credits - obj.debits < $userInput; } } );;
```

Diğer enjeksiyon türlerini test ederken olduğu gibi, bir sorunu göstermek için güvenlik açığından tam olarak yararlanmanıza gerek yoktur. Hedef API diliyle ilgili özel karakterler enjekte ederek ve
sonuçları gözlemleyerek, bir test cihazı uygulamanın girdiyi doğru şekilde ifade edip etmediğini belirleyebilir. Örneğin MongoDB içinde, aşağıdaki özel karakterlerden herhangi birini içeren bir dize sterilize edilmemiş olarak geçirilirse, bir veritabanı hatasını tetikler.

```
' " \ ; { }
```

Normal SQL enjeksiyonu ile benzer bir güvenlik açığı, bir saldırganın keyfi SQL komutlarını yürütmesine izin verir - verileri istediğiniz gibi ifşa eder veya manipüle

eder. Bununla birlikte, JavaScript tamamen öne çıkan bir dil olduğu için, bu sadece bir saldırganın verileri manipüle etmesine izin vermekle kalmaz, aynı zamanda keyfi kod çalıştırmaya da izin verir. Örneğin, test yaparken sadece bir hataya neden olmak yerine, tam bir istismar, geçerli JavaScript'i oluşturmak için özel karakterleri kullanır.

Bu giriş `0;var date=new Date(); do{curDate = new Date();}while(curDate-date<10000)` İçine yerleştirilir `$userInput` Yukarıdaki örnekte kod, aşağıdaki JavaScript işlevinin yürütülmesine neden
olacaktır. Bu özel saldırı dizisi, tüm MongoDB örneğinin 10 saniye boyunca% 100 CPU kullanımında olmasını sağlayacaktır.

`function() { return obj.credits - obj.debits < 0;var date=new Date(); do{curDate = new Date();}while(curDate-date<10000); }`

**Example 2 (Örnek 2)**

Sorgularda kullanılan giriş tamamen sterilize edilmiş veya parametrelendirilmiş olsa bile, NoSQL enjeksiyonunu tetikleyebileceği alternatif bir yol vardır. Birçok NoSQL örneğinin, uygulama programlama dilinden bağımsız kendi ayrılmış değişken isimleri vardır.

Örneğin MongoDB içinde, the `$where` Sintax'ın kendisi ayrılmış bir sorgu operatörüdür. Sorguya tam olarak gösterildiği gibi aktarılması gerekir; herhangi bir değişiklik bir veritabanı hatasına neden olur. Ancak, çünkü `$where` Ayrıca geçerli bir PHP değişken adıdır, bir saldırganın adı verilen bir PHP değişkeni oluşturarak sorguya kod eklemesi mümkün olabilir `$where` . . PHP MongoDB belgeleri geliştiricileri açıkça uyarıyor:

Lütfen tüm özel sorgu operatörleri için (başlayarak) emin olun `$` ) PHP'nin değiştirmeyi denememesi için tek teklifler kullanırsınız `$exists` Değişkenin değeri ile `$exists` . .

Bir sorgu, aşağıdaki örnek gibi hiçbir kullanıcı girdisine bağlı olmasa bile, bir saldırgan operatörü kötü amaçlı verilerle değiştirerek MongoDB'den yararlanabilir.

`db.myCollection.find( { $where: function() { return obj.credits - obj.debits < 0; } } );`

PHP değişkenlerine potansiyel olarak veri atamanın bir yolu HTTP Parametre Kirliliği ile (bkz: HTTP Parametre kirliliği için test). Adında bir değişken oluşturarak `$where` Parametre kirliliği yoluyla, sorgunun artık geçerli olmadığını gösteren bir MongoDB hatasını tetikleyebilir. Herhangi bir değer `$where` İpleme dışında

`$where` Kendisi, kırılganlığı göstermek için yeterli olmalıdır. Bir saldırgan aşağıdakileri ekleyerek tam bir istismar geliştirir:

`$where: function() { //arbitrary JavaScript here }`

## Referances (Referanslar)

## Injection Payloads (Enjeksiyon Yükleri)

- MongoDB için NoSQL Enjeksiyon örnekleri ile enjeksiyon yükü kelime listesi

**Whitepapers (Beyaz Kağıtlar)**

- Bryan Sullivan from Adobe: “Server-Side JavaScript Injection”
- Bryan Sullivan from Adobe: “NoSQL, But Even Less Security”
- Erlend from Bekk Consulting: “[Security] NOSQL-injection”
- Felipe Aragon from Syhunt: “NoSQL/SSJS Injection”
- MongoDB Documentation: “How does MongoDB address SQL or Query injection?”
- PHP Documentation: “MongoCollection::find”
- Hacking NodeJS and MongoDB
- Attacking NodeJS and MongoDB