Testing for MySQL (MySQL için Test)

Summary (Özet)

SQL Injection güvenlik açıkları, bir SQL sorgusunun yapımında yeterince kısıtlanmadan veya sterilize edilmeden kullanılırsa ortaya çıkar. Dinamik SQL kullanımı (swig sorgularının dizelerin oluşturulması) kullanımı bu güvenlik açıklarına kapı açar. SQL enjeksiyonu, bir saldırganın SQL sunucularına erişmesini sağlar. Kullanıcının veri tabanına bağlanmak için kullanılan ayrıcalıkları altında SQL kodunun yürütülmesine izin verir.

MySQL sunucusunun birkaç özelliği vardır, böylece bazı istismarların bu uygulama için özel olarak özel olarak özelleştirilmesi gerekir. Bu bölümün konusu budur.

How to Test (Nasıl Test Edilir)

Bir MySQL veritabanı tarafından desteklenen bir uygulamada bir SQL enjeksiyonu güvenlik açığı bulunduğunda, DBMS'deki MySQL sürümüne ve kullanıcı ayrıcalıklarına bağlı olarak gerçekleştirilebilecek bir dizi saldırı vardır.

MySQL, dünya çapında üretimde kullanılan en az dört versiyonla birlikte gelir. `3.23.x`, `4.0.x`, `4.1.x` ve `5.0.x`. . Her sürüm, sürüm numarasıyla orantılı bir dizi özelliğe sahiptir.

- Versiyon 4.0: UNION
- Versiyon 4.1'den: Alt Sulamalar
- Sürüm 5.0'dan: Saklanan prosedürler, Saklanan fonksiyonlar ve adı verilen görünüm `INFORMATION_SCHEMA`
- Sürüm 5.0.2: Tetikleyiciler

4.0.x'ten önceki MySQL sürümleri için, alt tipi işlevsellik veya subuçlu olduğundan, yalnızca Boolean veya zamana dayalı Kör Enjeksiyon saldırılarının kullanılabileceğine dikkat edilmelidir. `UNION` İfadeler uygulanmadı.

Bundan sonra, SQL Enjeksiyonu için Test Bölümünde açıklanana benzer bir istekle tetiklenebilen klasik bir SQL enjeksiyonu güvenlik açığı olduğunu varsayacağız.

`http://www.example.com/page.php?id=2`

The Single Quotes Problem (Tek Alıntı Sorunu)

MySQL özelliklerinden yararlanmadan önce, genellikle web uygulamaları tek alıntılardan kaçtığı için, bir ifadede dizelerin nasıl temsil edilebileceği göz önünde bulundurulmalıdır.

MySQL alıntısı aşağıdaki gibidir:

`'A string with \'quotes\''`

Yani, MySQL kaçan kesme işaretlerini yorumluyor `\'` Karakterler olarak değil, metakaraktörler olarak.

Yani, uygulamanın düzgün çalışması gerekiyorsa, sabit dizeler kullanması gerekiyorsa, iki vaka farklılaştırılmalıdır:

1. Web uygulaması tek tekliflerden kaçar `'` ⇒ `\'`
2. Web uygulaması tek alıntılardan kaçmıyor `'` ⇒ `'`

MySQL altında, tek tekliflere ihtiyaç duymadan ilan edilecek sürekli bir ipe sahip olan tek tekliflerin ihtiyacını atlamanın standart bir yolu vardır.

Diyelim ki adı geçen bir alanın değerini bilmek istiyoruz `password` Bir kayıtta, aşağıdaki gibi bir durumla:

1. şifresi gibi `'A%'`
2. ASCII, kapsanan bir hex'te değer verir: `password LIKE 0x4125`
3. Char() fonksiyonu: `password LIKE CHAR(65,37)`

Multiple Mixed Queries (Birden Fazla Karışık Sorgulama)

MySQL kütüphane konnektörleri, ayrılan birden fazla sorguyu desteklemez `;` Bu nedenle, Microsoft SQL Server gibi tek bir SQL enjeksiyonu güvenlik açığına birden fazla homojen olmayan SQL komutu enjekte etmenin bir yolu yoktur.

Örneğin, aşağıdaki enjeksiyon bir hataya neden olacaktır:

`1 ; update tablename set code='javascript code' where 1 --`

Information Gathering (Bilgi Toplaması)

Fingerprinting MySQL (Parmak izi MySQL)

Tabii ki, bilinmesi gereken ilk şey, bir arka uç veritabanı olarak MySQL DBMS olup olmadığıdır. MySQL sunucusu, diğer DBMS'lerin MySQL lehçesindeki bir maddeyi görmezden gelmesine izin vermek için kullanılan bir özelliğe sahiptir. Yorum engel olduğunda `'/**/'` Bir ünlem işareti içerir `'/*! sql here*/'` MySQL tarafından yorumlanır ve MySQL kılavuzunda açıklandığı gibi diğer DBMS tarafından normal bir yorum bloğu olarak kabul edilir.

Örnek:

```
1 /*! and 1=0 */
```

> MySQL mevcutsa, yorum bloğunun içindeki madde yorum şeklinde yorumlanacaktır.

Versiyon

Bu bilgiyi elde etmenin üç yolu vardır:

1. Küresel değişkeni kullanarak `@@version`
2. Fonksiyonu kullanarak VERSION()
3. Bir sürüm numarası ile yorum parmak izi kullanarak `/*!40110 and 1=0*/`

yani

```
if(version >= 4.1.10)
 add 'and 1=0' to the query.
```

Bunlar eşdeğerdir, çünkü sonuç aynıdır.

Bant enjeksiyonunda:

```
1 AND 1=0 UNION SELECT @@version /*
```

İnferansiyel enjeksiyon:

```
1 AND @@version like '4.0%'
```

Yanıt, satır satırlarına bir şey içerecektir:

```
5.0.22-log
```

Login User (Giriş Kullanıcısı)

MySQL Server'ın güvendiği iki tür kullanıcı vardır.

1. USER(): MySQL Server'a bağlı kullanıcı.
2. CURRENT_USER(): sorguyu yürüten dahili kullanıcı.

1 ile 2 arasında bir fark vardır. Ana olan, anonim bir kullanıcının herhangi bir isimle (izin verilirse) bağlanabilmesidir, ancak MySQL dahili kullanıcısı boş bir isimdir ('). Başka bir fark, depolanmış bir
prosedürün veya depolanmış bir işlevin, başka bir yerde beyan edilmediği takdirde yaratıcı kullanıcı olarak yürütülmesidir. Bu sayede kullanılabilir
`CURRENT_USER` . .

Bant enjeksiyonunda:

```
1 AND 1=0 UNION SELECT USER()
```

İnferansiyel enjeksiyon:

```
1 AND USER() like 'root%'
```

Yanıt, satır satırlarına bir şey içerecektir:

```
user@hostname
```

Database Name in Use (Veritabanı Adı Kullanımda)

Yerli fonksiyon var `DATABASE()`

Bant enjeksiyonunda:

```
1 AND 1=0 UNION SELECT DATABASE()
```

İnferansiyel enjeksiyon:

```
1 AND DATABASE() like 'db%'
```

> Beklenen Sonuç, böyle bir dize: `dbname`

INFORMATION_SCHEMA (BİLGİ_ŞEMASI)

MySQL 5.0'dan INFORMATION_SCHEMA adlı bir görüş oluşturuldu. Veritabanı, tablolar ve sütunların yanı sıra prosedürler ve işlevler hakkında tüm bilgileri almamızı sağlar.

Tables_in_INFORMATION_SCHEMA	DESCRIPTION
SCHEMATA	All databases the user has (at least) SELECT_priv (Kullanıcının sahip olduğu tüm veritabanlar (en azından) SEELECT_priv)
SCHEMA_PRIVILEGES	The privileges the user has for each DB (Kullanıcının her bir DB için sahip olduğu ayrıcalıklar)

TABLES	All tables the user has (at least) SELECT_priv (Kullanıcının sahip olduğu tüm tablolar (en azından) SEÇKİZİ_priv)
TABLE_PRIVILEGES	The privileges the user has for each table (Kullanıcının her bir masa için sahip olduğu ayrıcalıklar)
COLUMNS	All columns the user has (at least) SELECT_priv (Kullanıcının sahip olduğu tüm sütunlar (en azından) SEÇKİZİ_priv)
COLUMN_PRIVILEGES	The privileges the user has for each column (Kullanıcının her sütun için sahip olduğu ayrıcalıklar)
VIEWS	All columns the user has (at least) SELECT_priv (Kullanıcının sahip olduğu tüm sütunlar (en azından) SEÇKİZİ_priv)
ROUTINES	Procedures and functions (needs EXECUTE_priv) (Prosedürler ve fonksiyonlar (İnçersiz Kıymettir_prive))
TRIGGERS	Triggers (needs INSERT_priv) (Tetikleyiciler (İNSERT_prive'ye ihtiyaç duyar))
USER_PRIVILEGES	Privileges connected User has (Özelleştirilen Kullanıcıya Bağlı)

Tüm bu bilgiler, SQL Injection bölümünde açıklanan bilinen tekniklerin kullanılmasıyla çıkarılabilir.

Attack Vectors (Vektörlere saldırı)

Write in a File (Bir dosyada yazın)

Bağlı kullanıcı varsa `FILE` Ayrıcalıklar ve tek alıntılar kaçılmaz, `into outfile` Madde, sorgu sonuçlarını bir dosyada ihraç etmek için kullanılabilir.

```
Select * from table into outfile '/tmp/file'
```

Not: Bir dosya adını çevreleyen tek alıntıları atlamanın bir yolu yoktur. Yani kaçış gibi tek alıntılarda biraz sanitasyon varsa `\'` Kullanmanın bir yolu olmayacak `into outfile` Madde.

Bu tür bir saldırı, bir sorgunun sonuçları hakkında bilgi edinmek veya web sunucusu dizininde gerçekleştirilebilecek bir dosya yazmak için bant dışı bir teknik

olarak kullanılabilir.

Örnek:

```
1 limit 1 into outfile '/var/www/root/test.jsp' FIELDS ENCLOSED BY '//'  LINES TERMINATED BY '\n<%jsp code here%>';
```

> Sonuçlar bir dosyada saklanır rw-rw-rwMySQL kullanıcısı ve grubuna ait ayrıcalıklar.
>
> Nerede `/var/www/root/test.jsp` Şunları içerecektir:
>
> `//field values//<%jsp code here%>`

Read from a File (Bir dosyadan okuyun)

`load_file` Dosya sistemi izinleri ile izin verildiğinde bir dosyayı okuyabilen yerel bir işlevdir. Bağlı bir kullanıcı varsa `FILE` Ayrıcalıklar, dosyaların içeriğini almak için kullanılabilir. Tek alıntılar, daha
önce açıklanan teknikleri kullanarak atlayarak sanitasyondan kaçabilir.

```
load_file('filename')
```

> Tüm dosya, standart teknikler kullanılarak ihracat için kullanılabilir.

Standard SQL Injection Attack (Standart SQL Enjeksiyon Saldırısı)

Standart bir SQL enjeksiyonunda, sonuçları doğrudan bir sayfada normal çıktı veya MySQL hatası olarak görüntüleyebilirsiniz. Zaten bahsedilen SQL Enjeksiyon saldırılarını ve zaten açıklanan MySQL özelliklerini kullanarak, doğrudan SQL enjeksiyonu, öncelikle pentester'ın karşılaştığı MySQL sürümüne bağlı olarak bir seviye derinlikte kolayca gerçekleştirilebilir.

İyi bir saldırı, bir işlevi / prosedürü veya sunucunun kendisini bir hata atmaya zorlayarak sonuçları bilmektir. MySQL tarafından atılan hataların bir listesi ve özellikle yerel fonksiyonların MySQL Manual'da bulunabilir.

Out of Band SQL Injection (Grup SQL Enjeksiyonu Dışı)

Bant enjeksiyonu dışında, kullanılarak gerçekleştirilebilir `into outfile` Madde.

Blind SQL Injection (Kör SQL Enjeksiyonu)

Kör SQL enjeksiyonu için, MySQL sunucusu tarafından doğal olarak sağlanan bir dizi yararlı işlev vardır.

- Dize Uzunluğu:
  - `LENGTH(str)`
- Belirli bir ipten bir alt dizeyi çıkarın:
  - `SUBSTRING(string, offset, #chars_returned)`
- Zamana Dayalı Kör Enjeksiyon:
  - KARIŞIKLIĞI VE UYKU

    `BENCHMARK(#ofcycles,action_to_be_performed)` Kıyaslama fonksiyonu, boolean değerleri tarafından kör enjeksiyon herhangi birsonuç vermediğinde zamanlama atakları yapmak için kullanılabilir. Bak. `SLEEP()` (MySQL > 5.0.x) yedek kriterinde bir alternatif için.

Tam bir liste için, MySQL kılavuzuna bakın

Tools (Araçlar)

- Francois Laouche: Çoklu DBMS SQL Enjeksiyon aracı
- Reversing.org - sqlbttools
- Bernardo Damele A. G.: sqlmap, otomatik SQL enjeksiyon aracı
- Muhaim Dzulfakar: MySqloit, MySql Enjeksiyon alma aracı

Referance (Referanslar)

WhitePapers (Beyaz kağıtlar)

- Chris Anley: Hackproofing MySQL

Case Studies (Vaka Çalışmaları)

- Zeelock: MySQL Veritabanlarında Kör Enjeksiyon