


T.C.

BARTIN ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

MAHMUD MARDİNİ 18670310077

BSM323

Bilgisayar Güvenliğine Giriş

Dönem Projesi Raporu

Prepared Statements Automator

1. Giriş

Günümzde yeni teknolojiler gellişmesiyle ve teknoloji ürünleri artmasıyla, siber suçlar da artmaktadır. Siber saldırılar oldukça çeşitlidir. Bazı sldırganlar, bir sistemin çöktürmeye çalışırlar bazıları bir kişinin bilgileri çalmak isterler bazıları da bir sistemdeki var olan bilgiyi değiştirmek isterlerdir. Bu saldırılarda çeşitli yöntemleri kullanılmaktadır. Bu yöntemlerin en yaygın ve en tehlikelilerden biri olan SQL Enjeksiyonudur. SQL Enjeksiyonnu Yanlış kodlanmış web formları istismar edilir. Formda beklenen veriler yerine kötü niyetli SQL komutlarını girilerek, sistmin veritabnındaki bilgiler üzerinde veri değiştirme, veri silme ve yetkisiz erişim sağlama gibi çeşitli işlemler yapılabilmektedir.

SQL Enjiksiyonu tehlikesinden korunmak için en önemli ve en etkin yöntem ise güvenli kod yazmaktır. Bu tür saldırılardan korunmak amacıyla projemiz geliştirilmiştir.

2. Projenin Amacı ve Faydaları:

- Uygulamarın SQL Enjeksiyonu açıklıkları kapatılması
- Bir sistemin veritabanındaki bilgileri korunması
- Güvenli olmayan SQL kodları düzenlemesi

3. Problenin tanımlaması:

SQL sorguları yaparken çeşitli ifadeleri kullanarak yapılabilmektedir.

PHP programlama dilinde bir örnek verelim: diyelimki veritabanında users adlı bir tablomuz var. Bu tablo, kullanıcıların isimleri, epostaları, şifreleri gibi bilgiler içermektedir. Kullanıcıların sisteme giriş yapmayı sağlamak için aşağdaki SQL ifadeleri gibi kullanılır.

$statment = çekilen bilgilerin değişkeni.

$email: kullanıcı tarafından girilen eposta.

$password: kullanıcı tarafından girilen şifre.

- $pdo->query(“SELECT * FROM users WHERE email = ‘$email’ AND password = ‘$password’ ”);
- $statement = $pdo->query(“SELECT * FROM users WHERE email = ‘$email’ AND password = ‘$password’ ”);
- $statement = $pdo->prepare(“SELECT * FROM users WHERE email = ‘$email’ AND password = ‘$password’ ”);

$statement->execute();

- $statement = $pdo->prepare(“SELECT * FROM users WHERE email = :email AND password = :password ”);

$statement->execute(array(':email'=>$email, ':password'=>$password));

Yukarıdaki belirtilen SQL ifadelerin en güvenlisisi ise son ifadedir. Çünkü 1., 2. ve 3. ifadelerde, kullanıcı tarafından girilen veriler direk SQL ifadesiyle birleştirilmiştir. Yani girilen veriler SQL kodları içeriyorsa diğer yazılan kodlar ile çalıştırılacaktır. Bazı yazılımcılar HTML5 kullanarak input filtreleme yaptıkları için çalıştıkları uygulama güvenli olduğunu varsayıyorlar ama HTML kodları browser’da görülmeye ve değiştirilmeye açık olduğu için yapılan filtreleme de kaldırılabilmektedir. Bu nedenle server-side kodları filttrelemesi kesinlikle yapılması gereken şeydir. Bu filtrlemenin yanında da 4. ifade gibi doğru prepared statement syntax’i kullanılılrsa uygulama SQL Enjeksiyonundan güvenceye alınmış olur.

4. Proje gerçekleştirme adımları:

Girilen kodları satır satır parçalanarak ve satırları analiz ederek güvenli olmayan SQL ifadeleri tespit ettikten ve bu ifadelerde kullanılan kullanıcı tarafından girilen bilgileri içeren değişkenleri belirledikten sonra, ifadenin türüne göre ilgili algoritma çalıştırılıyor. Diyelim ki uygulamayı yapan kişi aşağıdaki ifade kullandı.

$pdo->query(“SELECT * FROM users WHERE email = ‘$email’ AND password = ‘$password’ ”);

İfadeyi güvenli olmadığını tespit ettikten sonra, ifadenin değişkeni olup olmadığını kontrol edilir. Değişkeni olmadığını tespit ettikten sonra php prepared statement syntax’a uygun olarak yapılması için ifadeye otomatik olarak bir değişken atanır ($stmt0). Değişkeni olmayan ifadelerin sayısına göre değişkenin ismi değiştirilir ($stmt1, $stmt2 vs.) ve “query” ifadesi “prepare” ile değiştirilir.

Yani ifademiz bu şekilde oldu:

$stmt0 = $pdo->prepare(“SELECT * FROM users WHERE email = ‘$email’ AND password = ‘$password’ ”);

Daha sonra değişkenleri tutmak için kullanılan simge tespit ediliyor (tek veya çif tırnak işareti) ve simgelerin arasındaki uzunluklara göre değişkenleri alınıp bir array'a atanıyor. Değişkenleri aldıktan sonra da SQL ifadesinde değişkenleri tanımşayan simgeler(tırnak işareti) kaldırılıyor ve değişkenlerin başındaki dolar işareti ($) yerine iki nokta (:) yerleştiriliyor.
Yeni ifademiz:
$stmt0 = $pdo->prepare("SELECT * FROM users WHERE email = :email AND password = :password ");
Son aşamada yeni satır oluşturulup bu satıra "ifadenin_değişkeni -> execute();" eklenir.
Execute parantezleri içinde array() ekliniyor ve onun içinde tespit edilen değişkenlerin sayısına uygun olarak aşağıdaki ifade eklenir. ':değişken_adı' => $değişken_adı
Sonuç olarak elde ettiğimiz yeni SQL İfade:
$stmt0 = $pdo->prepare("SELECT * FROM users WHERE email = :email AND password = :password ");
$stmt0 ->execute(array(':email'=>$email, ':password'=>$password));

Bu şekilde projedeki diğer kodlara mudahale etmeden sadece güvenli olmayan kodları düzenleniyor.
Yeni oluşturulan SQL ifademiz PHP Prepared Statements syntax'a uygun olarak çalışıyor ve artık kullanıcı tarafından girilen veriler herhangi bir SQL komutu içeriyorsa bile öncelikle bir array'a alınır ve daha sonra veritabanına bir SQL komutu olarak değilde bir String olarak gönderilir.

5. Sonuç:

Yazılım sahipleri kendi veritabanı sorgulamaları yapan kodları tek tek incelemektense, koda herhangi bir müdahalede bulunmadan SQL Enjeksiyonu yöntemi kullanan saldırılardan çok basit bir işlem yaparak korunabilmektedir. Bu sayede, yazılımcılara zaman, kolaylık ve güvenlik sağlayan ve önceden yapılmayan bir yazılım aracı geliştirilmiş oldu.

6. Görüntüler: