Django ve Güvenlik

Emre Yılmaz, Ekim 2009, WGT E-Dergi 2. Sayı

Giriş

Django, Python ile yazılmış bir web geliştirme çatısıdır. Esnekliği, kolaylığı, hızı, geliştirme hızını çok düşük zamanlara çekmesi ve Python'ın gücünü de ekleyince ileride adını daha da çok duyacağımız bir çatı. Dışarıdan durum gayet güzel görünüyor ama "security ninja" iseniz aklınıza hemen şu soru geliyor: "Peki ya güvenlik?"

Uygulama katmanına genel bakış

Kullanıcıdan aldığınız veri şeytandır! Hiç bir şekilde tarayıcıdan aldığınız veriye güvenemezsiniz. Direkt olarak alıp uygulamanızda kullanamazsınız, ilgili veriyi işlemeden önce gereken kontroller (veri tipi kontrolü, filtreleme vs.) kesinlikle yapılmalıdır.

Peki, "Django bu noktada neler yapıyor, geliştiriciye neler sağlıyor?" genel olarak web uygulamalarında ortaya çıkan güvenlik zaafiyetlerinin isimleriyle başlıklar halinde görelim.

XSS

Django'nun template motoru ekrana basılan değişkenleri otomatik olarak "escape" ediyor. Adres satırından bir veri alıp template dosyanızda direkt olarak kullansanız bile XSS diye bir problem kalmıyor. Örneğin adres satırı üstünden veri alıp basitçe ekrana basan bir template ve view çağrısı yazalım:

views.py

```
def xssTest(request):
  deger = request.GET.get('deger')
  return render_to_response("test.tpl", {"deger": deger},
  context_instance=RequestContext(request))
```

test.tpl

```
Django <strong>{{deger}}</strong>
```

urls.py dosyasına da urlpatterns değişkenine (r'^test/', xssTest) şeklinde bir ifade eklediğinizde artık adres/test adresinden son durumu görmek mümkün olacaktır. "test/?deger=sihirli+bir+sey" gibi bir istek yaparsak şöyle bir çıktı karşımıza gelecek:

```
Django <strong>sihirli bir sey</strong>
```

Veri adres satırından geliyor, peki işin içine biraz JavaScript karıştırırsak? test/?deger=<script>alert(1);</script> için şöyle bir sonuçla karşılaşıyoruz:

Django <script>alert(1)

Verilerinizi ekrana basmadan önce Django template motorunu kullandığınız sürece XSS konusunda dert etmenize gerek yok, zaten tüm değişkenler ekrana basılmadan önce gerekli güvenlik kontrolleri yapılıyor. (Django harika bir şey olduğu için ilgili methodun üstüne yazıp kendi escape fonksiyonunu yazabilir ya da bazı değişkenler için bu durumu iptal edebilir [1], istediğiniz şekilde kullanabilirsiniz. Bu konuda gerekli esneklik sağlanıyor.)

SQL Injection

Django ile uygulama geliştirirken SQL Injection konusunda da dert etmenize gerek yok. Zira veritabanı sürücüleri otomatik olarak gerekli kontrolleri yapıyor. Kullandığınız veritabanı türüne göre (PostgreSQL, mySQL, Oracle vb.) otomatik olarak uygun karakterlerle filtrelemeler ve escape işlemleri yapılıyor. Tabi veritabanı işlemlerinde Django'nun getirdiği veritabanı API'sini kullanmak şartıyla.

Bir alt seviyeye inip sorguları elle yazıp, elle işlerseniz haliyle o zaman sorguya kullanıcıdan bir veri ekleyeceğiniz zaman gerekli kontrolleri yapmak zorundasınız. Gerekli escape işlemleri için yine Django'dan gerekli fonksiyonları kullanabilirsiniz. Örneğin; escape etmek için django.connection.ops.quote_name fonksiyonunu kullanabilirsiniz.

Django'nun veritabani sürücüsü yeterince güçlü. Kendiniz elle sorgu yazmak zorunda kalmayacaksanız çok komplike işler yapmadığınız sürece. Ama yaptığınızda haliyle genel olarak SQL Injection'a karşı önlemleri geliştiricinin alması gerekiyor.

CSRF

Django, CSRF saldırılarına karşı oldukça kullanışlı bir ara uygulama (middleware) sağlıyor. Projenizin ayar dosyasında (settings.py) middleware kısmına django.contrib.csrf.middleware.CsrfMiddleware satırını eklediğinizde otomatik olarak CSRF saldırılarına karşı formlarınız korunmuş hale geliyor.

CSRF middleware, tüm formlara gizli bir input ekleyip, rastgele bir değerle dolduruyor, aynı değeri oturuma kayıt ediyor ve formun işlendiği yerde bu ikisini kontrol ediyor. GET istekleri için herhangi bir koruma mevcut değil ama zaten GET üstünden veri alıp oturuma dair işlemler yapmak genel güvenlik anlayışında kesinlikle karşı çıkılan bir durum. Zira W3C tarafından hazırlanan resmi HTTP 1/1 metod tanımlamalarına dahi baktığınızda GET'in bu tip işlemler için güvensiz olduğu belirtilmiştir [2].

Yazdığınız ajax uygulamalarında ise, bir sunucu değişkeni olarak X-Requested-With: XMLHttpRequest dönecektir. Eğer bu değişken varsa Django CSRF korumasını devre dışına alıyor. Zira, bu sadece ajax isteği yapıldığı zaman dönmekte ve CSRF için bir tehlike arz etmemektedir.

Diğer Saldırı Türleri

E-mail/Header injection konusunda Django'nun standart fonksiyonları gereksiz karakterleri filtreliyor.

Sonuç

Django, güvenlik açısından geliştiriciye ciddi anlamda kolaylık sağlıyor, fakat sistemin her noktasına müdahil olamayacağı için iş geliştiriciye de düşüyor. Genel güvenlik ilkeleri doğrultusunda hazırlandığı sürece herhangi bir sıkıntı ortaya çıkmayacaktır. Uygulama katmanın dışına çıkıp biraz daha alt seviyeye -Django'nun çekirdeğine- inersek, Django'nun, baştan itibaren güvenlik ilkelerine göre yazılmış ve her adımda gerekli güvenlik incelemeleri yapılmış bir web geliştirme çatısı demek iddialı bir cümle olabilir. Hatta bildirilen güvenlik açıklarının kaybolması, geri dönülmemesi gibi sıkıntılar da yaşandı geçmişte şu aralar düzelse de.

Her uygulama gibi Django'da da belirli aralıklarla güvenlik açıkları ortaya çıkıyor ve kısa zamanda kapatılıyor. Nasıl python'da, apache'de güvenlik açığı çıkıyor ve kısa süre içinde kapatılıyorsa bunu da benzer bir durum olarak görebilirsiniz. Bu noktada uygulama geliştiricisinin düzenli olarak Django projesini takip edip, olası güncellemeleri sistemine adapte etmesi yeterli olacaktır.

Referanslar

[1] http://docs.djangoproject.com/en/dev/ref/templates/builtins/
[2] http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

Kaynaklar

http://www.djangobook.com/en/beta/chapter20/
http://docs.djangoproject.com/en/dev/ref/templates/builtins/#autoescape
http://docs.djangoproject.com/en/dev/ref/contrib/csrf/