Testing for HTTP Parameter Pollution (HTTP parametre Kirliliği için test)

Summary (Özet)

HTTP Parametre Kirliliği, aynı adı taşıyan birden fazla HTTP parametresi almaya yönelik başvuru yanıtını test eder; örneğin, parametre varsa `username` GET veya POST parametrelerine iki kez dahildir.

Birden fazla HTTP parametresinin aynı adı taşıyan bir şekilde kullanılması, değerleri beklenmedik şekillerde yorumlamak için bir başvuruya neden olabilir. Bu etkilerden yararlanarak, bir saldırgan giriş doğrulamasını atlayabilir, uygulama hatalarını tetikleyebilir veya iç değişken değerlerini değiştirebilir. HTTP Parametre Kirliliği (kısa
HPP) tüm web teknolojilerinin bir yapı taşını etkilediğinden, sunucu ve istemci tarafı saldırıları vardır.

Mevcut HTTP standartları, birden fazla giriş parametresinin aynı adı
ile nasıl yorumlanacağı konusunda rehberlik içermez. Örneğin, RFC 3986,
Query String terimini birdizi alan değeri çifti olarak tanımlar ve RFC 2396, ters çevrilmiş ve rezerve edilmemiş sorgu dize karakterleri sınıflarını tanımlar. Bir standart olmadan, web uygulama bileşenleri bu kenarı
kılıfı çeşitli şekillerde ele alır (ayrıntılar için aşağıdaki tabloya bakın).

Kendi başına, bu mutlaka bir güvenlik açığının göstergesi değildir. Bununla birlikte, geliştirici sorunun farkında değilse, çoğaltılmış parametrelerin varlığı, uygulamada bir saldırgan tarafından potansiyel olarak istismar edilebilecek anormal bir davranış üretebilir. Güvenlikte sık sık olduğu gibi, beklenmedik davranışlar bu durumda HTTP Parametre Kirliliği saldırılarına yol açabilecek olağan bir zayıflık kaynağıdır. Bu güvenlik açığı sınıfını ve HPP saldırılarının sonucunu daha iyi tanıtmak için, geçmişte keşfedilen bazı gerçek yaşam örneklerini analiz etmek ilginçtir.

Input Validation and Filters Bypass (Giriş doğrulama ve filtreler Bypass)

2009 yılında, HTTP Parametre Kirliliği üzerine ilk araştırmanın yayınlanmasından hemen sonra, teknik, web uygulaması güvenlik duvarlarını atlamanın olası bir yolu olarak güvenlik topluluğundan ilgi gördü.

ModSecurity SQL Injection Core Rules'u etkileyen bu kusurlardan biri, uygulamalar ve filtreler arasındaki empedans uyumsuzluğunun mükemmel bir örneğini temsil eder. ModSerüs filtresi aşağıdaki dize için bir inkar listesi doğru bir şekilde uygulayacaktır:
`select 1,2,3 from table` Böylece bu örnek URL'nin web sunucusu tarafından işlenmesini engellemek:
`/index.aspx?page=select 1,2,3 from table`. . Bununla birlikte, birden fazla HTTP parametresinin birleştirilmesinden yararlanarak, bir saldırgan, ModSecurity filtresini girdiyi kabul etmesinden sonra uygulama sunucusunun ipi bir araya getirmesine neden olabilir. Örnek olarak, URL

`/index.aspx?page=select 1&page=2,3` Masadan itibaren ModSecurity filtresini tetiklemeyecektir, ancak uygulama katmanı girişin tam kötü niyetli ipe geri gelmesini sağlayacaktır.

Bir başka HPP güvenlik açığı, birçok UNIX sistemi tarafından kullanılan iyi bilinen baskı sistemi *olan Apple Cups'ı* etkiledi. HPP'yi ıstır kıvıran bir saldırgan, aşağıdaki URL'yi kullanarak bir Çapraz Site Senaryocu güvenlik açığını kolayca tetikleyebilir: `http://127.0.0.1:631/admin/?kerberos=onmouseover=alert(1)&kerberos` . . Uygulama doğrulama kontrol noktası ekstra ekleyerek atlanabilir `kerberos` Geçerli bir ipe sahip argüman (ör. örneğin boş dize). Doğrulama kontrol noktası sadece ikinci olayı göz önünde bulunduracağından, ilki `kerberos` Parametre, dinamik HTML içeriği oluşturmak için kullanılmadan önce düzgün bir şekilde sterilize edilmedi. Başarılı sömürü, barındırma web sitesi bağlamında JavaScript kodu uygulamasıyla sonuçlanacaktır.

Authentication Bypass (Kimlik doğrulama baypas)

Popüler blog platformu *Blogger'da* daha da kritik bir HPP güvenlik açığı keşfedildi. Hata, kötü niyetli kullanıcıların aşağıdaki HTTP talebini kullanarak kurbanın blogunun mülkiyetini almasına izin
verdi (`https://www.blogger.com/add-authors.do`) :: `POST /add-authors.do HTTP/1.1[...] security_token=attackertoken&blogID=attackerblogidvalue&blogID=victimblogidvalue&authorsList=goldshlager19test%40gmail.com(attacker email)&ok=Invite`

Kusur, ilkinde güvenlik kontrolü yapıldığı için web uygulaması tarafından kullanılan kimlik doğrulama mekanizmasında yer aldı. `blogID` parametre, gerçek operasyon ise ikinci olayı kullandı.

Expected Behavior by Application Server (Uygulama Sunucusu tarafından beklenen davranış)

Aşağıdaki tablo, farklı web teknolojilerinin aynı HTTP parametresinin birden fazla oluşumunun varlığında nasıl davrandığını göstermektedir.

URL ve sorgular göz önüne alındığında: `http://example.com/?color=red&color=blue`

Web Application Server Backend	Parsing Result	Example
ASP.NET / IIS	All occurrences concatenated with a comma	color=red,blue
ASP / IIS	All occurrences concatenated with a comma	color=red,blue
PHP / Apache	Last occurrence only	color=blue
PHP / Zeus	Last occurrence only	color=blue
JSP, Servlet / Apache Tomcat	First occurrence only	color=red
JSP, Servlet / Oracle Application Server 10g	First occurrence only	color=red
JSP, Servlet / Jetty	First occurrence only	color=red
IBM Lotus Domino	Last occurrence only	color=blue

IBM HTTP Server	First occurrence only	color=red
mod_perl, libapreq2 / Apache	First occurrence only	color=red
Perl CGI / Apache	First occurrence only	color=red
mod_wsgi (Python) / Apache	First occurrence only	color=red
Python / Zope	All occurrences in List data type	color= ['red','blue']

(Kaynak: Appsec AB 2009 Carettoni & Paola)

Test Objectives (Test Hedefleri)

- Sırt ucunu ve kullanılan ayrıştırma yöntemini belirleyin.
- Enjeksiyon noktalarını değerlendirin ve HPP kullanarak giriş filtrelerini atlamayı deneyin.

How to Test (Nasıl Test Edilir)

Neyse ki, HTTP parametrelerinin atanması tipik olarak web uygulama sunucusu aracılığıyla ele alındığından ve uygulama kodunun kendisi değil, parametre kirliliğine yanıtı test etmek tüm sayfalarda ve eylemler arasında standart olmalıdır. Bununla birlikte, derinlemesine iş mantığı bilgisi gerekli olduğundan, HPP'nin test edilmesi manuel test gerektirir. Otomatik araçlar, çok fazla yanlış pozitif üretme eğiliminde oldukları için denetçilere yalnızca kısmen yardımcı olabilir. Buna ek
olarak, HPP istemci tarafı ve sunucu tarafı bileşenlerinde kendini gösterebilir.

Server-Side HPP (Sunucu-Side HPP)

HPP güvenlik açıklarını test etmek için, kullanıcı tarafından sağlanan girişe izin veren herhangi bir formu veya eylemi belirleyin. HTTP GET isteklerindeki sorgu dize parametreleri tarayıcının gezinme
çubuğunda ince ayar yapmak kolaydır. Form eylemi POST aracılığıyla veri gönderirse, test cihazının sunucuya gönderildiği gibi POST verilerini kurcalamak için bir araya gelen bir proxy kullanması gerekir. Test etmek için belirli bir giriş parametresi belirledikten sonra, talebi durdurarak GET veya POST verilerini düzenleyebilir veya yanıt sayfası yüklendikten sonra sorgu dizisini değiştirebilir. HP güvenlik açıklarını test etmek, aynı parametreyi GET veya POST verilerine ekler, ancak
farklı bir değerle atanır.

Örneğin: test ederse `search_string` Sorgu dizesinde parametre, istek URL'si bu parametre adını ve değerini içerecektir:

`http://example.com/?search_string=kittens`

Belirli parametre diğer birkaç parametre arasında gizlenebilir, ancak yaklaşım aynıdır; diğer parametreleri yerinde bırakın ve yinelenenleri ekleyin:

`http://example.com/?mode=guest&search_string=kittens&num_results=100`

Aynı parametreyi farklı bir değerle ekleyin:

`http://example.com/?mode=guest&search_string=kittens&num_results=100&search_string=puppies`

Ve yeni isteği ilet.

Hangi değerlerin ayrıştırıldığını belirlemek için yanıt sayfasını analiz edin. Yukarıdaki örnekte, arama sonuçları gösterebilir `kittens`, `puppies`, her ikisinin bir kombinasyonu (`kittens,puppies` ya da `kittens~puppies` ya da `['kittens','puppies']`), boş bir sonuç verebilir veya hata sayfası.

Bu davranışın, ilk, son veya aynı adı taşıyan giriş parametrelerinin kombinasyonunu kullanmak, tüm uygulama boyunca tutarlı olması muhtemeldir. Bu varsayılan davranışın potansiyel bir güvenlik açığını ortaya çıkarıp oluşturmadığı, belirli bir uygulamaya özgü belirli giriş doğrulamasına ve filtrelemeye bağlıdır. Genel bir kural olarak: mevcut girdi doğrulaması ve diğer güvenlik mekanizmaları tek girişlerde yeterliyse ve sunucu sadece ilk veya son kirli parametreleri atarsa,
parametre kirliliği bir güvenlik açığı ortaya çıkarmaz. Yinelenen parametreler kapsanıyorsa, farklı web uygulama bileşenleri farklı olaylar kullanır veya testler bir hata oluştururursa, güvenlik açıklarını tetiklemek için parametre kirliliğini kullanabilme olasılığı artmaktadır.

Daha derinlemesine bir analiz, her bir HTTP parametresi için üç HTTP isteği gerektirecektir:

1. Standart parametre adı ve değerini içeren bir HTTP isteği gönderin ve HTTP yanıtını kaydedin. - Örneğin. `page?par1=val1`
2. Parametre değerini kurcalanmış bir değerle değiştirin, HTTP yanıtını gönderin ve kaydedin. - Örneğin. `page?par1=HPP_TEST1`
3. Adım (1) ve (2) bir araya getiren yeni bir talep gönderin. Yine, HTTP yanıtını kaydedin. - Örneğin. `page?par1=val1&par1=HPP_TEST1`
4. Önceki tüm adımlar sırasında elde edilen yanıtları karşılaştırın.(3) yanıt (1)'den farklıysa ve (3)'ten gelen yanıt da (2)'den farklıysa, sonunda HPP güvenlik açıklarını tetiklemek için kötüye kullanılabilen bir emsal uyumsuzluk vardır.

Bir parametre kirliliği zayıflığından tam bir istismar hazırlamak, bu metnin kapsamının ötesindedir. Örnekler ve detaylar için referanslara bakın.

Client-Side HPP (Müşteri-Yargı HPP)

Sunucu tarafı HPP'ye benzer şekilde, manuel test, istemci tarafı bileşenlerini etkileyen parametre kirliliği güvenlik açıklarını tespit etmek için web uygulamalarını denetlemek için tek güvenilir tekniktir.
Sunucu tarafındaki varyantta saldırgan, korunan verilere erişmek veya izin verilmeyen veya yürütülmesi gerekmeyen eylemleri gerçekleştirmek için savunmasız bir web uygulamasından yararlanırken, istemci tarafındaki saldırılar istemci tarafı bileşenlerini ve teknolojileri yıkmayı amaçlamaktadır.

HP istemci tarafı güvenlik açıklarını test etmek için, kullanıcı girişine izin veren herhangi bir formu veya eylemi tanımlamak ve bu girdinin bir sonucunu kullanıcıya geri döndürür. Bir arama sayfası

idealdir, ancak bir giriş kutusu çalışmayabilir (kazaya geçersiz bir kullanıcı adı gösteremediği için).

Sunucu tarafı HPP'ye benzer şekilde, her HTTP parametresini kirletin `%26HPP_TEST` ve kullanıcı tarafından sağlanan yükün *url kodlanmış* olaylarını arayın:

- `&HPP_TEST`
- `&HPP_TEST`
- vs.

Özellikle, içinde HPP vektörlerinin bulunduğu yanıtlara dikkat edin `data`, `src`, `href` Atıflar veya formlar eylemler. Yine, bu varsayılan davranışın potansiyel bir güvenlik açığını ortaya çıkarıp çıkarmadığı, belirli girdi doğrulama, filtreleme ve uygulama iş mantığına bağlıdır. Buna ek olarak, bu
güvenlik açığının XMLHttpRequest (XHR), çalışma zamanı özellik oluşturma ve diğer eklenti teknolojilerinde (örneğin) kullanılan sorgu dize parametrelerini de etkileyebileceğini fark etmek önemlidir. Adobe Flash'ın flaş değişkenleri).

Tools (Araçlar)

- OWASP ZAP Pasif / Aktif Tarayıcılar

Referances (Referanslar)

WhitePapers (Beyaz kağıtlar)

- HTTP Parametre Kirliliği - Luca Carettoni, Stefano di Paola
- Müşteri-yanlış-yagram Parametre Kirliliği Örneği (Yahoo! Klasik Posta kusuru) - Stefano di Paola
- HTTP parametresi Kirlilik Saldırıları Nasıl Tespit Edilir - Chrysostomos Daniel
- CAPEC-460: HTTP Parametre Kirliliği (HPP) - Evgeny Lebanidze
- Web Uygulamalarında Parametre Kirliliği Güvenlik Açıklarının Otomatik Keşfi - Marco Balduzzi, Carmen Torrano Gimenez, Davide Balzarotti, Engin Kirda