Testing Directory Traversal File Include (Dizin Çaprazlama Dosya Dahil Etme Testi)

Summary (Özet)

Birçok web uygulaması günlük işlemlerinin bir parçası olarak dosyaları kullanır ve yönetir. İyi tasarlanmış veya dağıtılmamış giriş doğrulama yöntemlerini kullanarak, bir saldırgan erişilebilir olması amaçlanmayan dosyaları okumak veya yazmak için sistemi kullanabilir. Özellikle durumlarda, keyfi kod veya sistem komutlarını yürütmek mümkün olabilir.

Geleneksel olarak, web sunucuları ve web uygulamaları, dosyalara ve kaynaklara erişimi kontrol etmek için kimlik doğrulama mekanizmaları uygular. Web sunucuları, kullanıcıların dosyalarını dosyalarını dosyayı dosyayı, dosya sistemindeki fiziksel bir dizilini temsil eden bir “köklü dizin” veya “web belge kökü” içinde sınırlamayı çalışır. Kullanıcılar bu dizini web uygulamasının hiyerarşik yapısının temel dizini olarak düşünmek zorundadır.

Ayrıcalıkların tanımı, hangi kullanıcıların veya grupların sunucuda belirli bir dosyaya erişebilmeleri, değiştirebilmeleri veya yürütmeleri gerektiğini belirleyen Erişim Kontrol Listeleri (ACL) kullanılarak
yapılır. Bu mekanizmalar, kötü niyetli kullanıcıların hassas dosyalara erişmesini önlemek için tasarlanmıştır (örneğin, ortak olan
`/etc/passwd` UNIX benzeri bir platformda dosyalayın) veya sistem komutlarının uygulanmasından kaçınmak için.

Birçok web uygulaması, farklı türde dosyaları içerecek şekilde sunucu tarafı komut dosyalarını kullanır. Bu yöntemi görüntüleri, şablonları, statik metinleri yüklemek ve benzeri şeyleri yönetmek için kullanmak oldukça yaygındır. Ne yazık ki, bu uygulamalar giriş parametreleri (yani, form parametreleri, çerez değerleri) doğru bir şekilde doğrulanmazsa güvenlik açıklarını ortaya koymaktadır.

Web sunucularında ve web uygulamalarında, bu tür bir sorun yol geçişi / dosyada saldırılar içerir. Bu tür bir güvenlik açığından yararlanarak, bir saldırgan normalde okuyamadıkları dizinleri veya

dosyaları okuyabilir, web belge kökü dışındaki verilere erişebilir veya harici web sitelerinden komut dosyaları ve diğer dosyaları içerebilir.

OWASP Test Kılavuzu'nun amacı için, yalnızca web uygulamalarıyla ilgili güvenlik tehditleri dikkate alınmayacak ve web sunucularına yönelik tehditler dikkate alınmayacaktır (örneğin, rezil `%5c` Microsoft IIS web sunucusuna kaçış kodu). İlgili okuyucular için referanslar bölümünde daha fazla okuma önerileri verilecektir.

Bu tür bir saldırı da nokta-nokta-slash saldırısı olarak bilinir (`../`), dizin geçişi, dizin tırmanma veya geri izleme.

Bir değerlendirme sırasında, yol geçişini ve dosyanın kusurları içerdiğini keşfetmek için testçilerin iki farklı aşamayı gerçekleştirmesi gerekir:

1. Giriş Vektörleri Yutma (her giriş vektörünün sistematik bir değerlendirmesi)
2. Test Teknikleri (bir saldırgan tarafından güvenlik açığını kullanmak için kullanılan her saldırı tekniğinin metodik bir değerlendirilmesi)

Test Objectives (Test Hedefleri)

- Yol geçişi ile ilgili enjeksiyon noktalarını belirleyin.
- Atlama tekniklerini değerlendirir ve yol geçişinin boyutunu tanımlar.

How to Test (Nasıl Test Edilir)

Black-Box Testing (Siyah-Kutu Testi)

Input Vectors Enumeration (Giriş Vektörleri Numaralandırma)

Uygulamanın hangi bölümünün giriş doğrulama bypass edilmesine karşı savunmasız olduğunu belirlemek için test cihazının, uygulamanın kullanıcıdan içerik kabul eden tüm bölümlerini numaralandırması gerekir. Bu aynı zamanda HTTP GET ve POST sorgularını ve dosya yüklemeleri ve HTML formları gibi ortak seçenekleri de içerir.

İşte bu aşamada gerçekleştirilecek kontrollerin bazı örnekleri:

- Dosyayla ilgili işlemler için kullanılabilecek talep parametreleri var mı?
- Olağandışı dosya uzantıları var mı?
- İlginç değişken isimler var mı?

- `http://example.com/getUserProfile.jsp?item=ikki.html`
- `http://example.com/index.php?file=content`
- `http://example.com/main.cgi?home=index.htm`

- Web uygulaması tarafından kullanılan çerezlerin dinamik nesil sayfaları veya şablonlar için tanımlanması mümkün mü?
  - `Cookie: ID=d9ccd3f4f9f18cc1:TM=2166255468:LM=1162655568:S=3cFpqbJgMSSPKVMV:TEMPLATE=flower`
  - `Cookie: USER=1826cc8f:PSTYLE=GreenDotRed`

Testing Techniques (Test Teknikleri)

Testin bir sonraki aşaması, web uygulamasında bulunan giriş doğrulama işlevlerini analiz etmektir. Önceki örneği kullanarak, dinamik sayfa adı `getUserProfile.jsp` Bir dosyadan statik bilgileri yükler ve içeriği kullanıcılara gösterir. Bir saldırgan kötü niyetli ipi ekleyebilir `../../../../etc/passwd` Bir Linux / UNIX sisteminin şifre hash dosyasını eklemek için. Açıkçası, bu tür bir saldırı ancak doğrulama kontrol noktası başarısız olursa mümkündür; dosya sistemi ayrıcalıklarına göre, web uygulamasının kendisinin dosyayı okuyabilmesi gerekir.

Bu kusuru başarılı bir şekilde test etmek için, testçinin test edilen sistemin ve istenen dosyaların bulunduğu hakkında bilgi sahibi olması gerekir. İstemenin bir anlamı yok `/etc/passwd` Bir IIS web sunucusundan.

`http://example.com/getUserProfile.jsp?item=../../../../etc/passwd`

Kurabiyeler için örnek:

`Cookie: USER=1826cc8f:PSTYLE=../../../../etc/passwd`

Dış web sitesinde bulunan dosyaları ve komut dosyaları da dahil olmak mümkündür:

`http://example.com/index.php?file=http://www.owasp.org/malicioustxt`

Protokoller argüman olarak kabul edilirse, yukarıdaki örnekte olduğu gibi, yerel dosya sistemini bu şekilde araştırmak da mümkündür:

`http://example.com/index.php?file=file:///etc/passwd`

Protokoller, yukarıdaki örneklerde olduğu gibi argüman olarak kabul edilirse, yerel hizmetleri ve yakın hizmetleri araştırmak da mümkündür:

`http://example.com/index.php?file=http://localhost:8080`
`http://example.com/index.php?file=http://192.168.0.2:9080`

Aşağıdaki örnek, herhangi bir yol geçiş karakteri kullanmadan bir CGI bileşeninin kaynak kodunu göstermenin nasıl mümkün olduğunu gösterecektir.

`http://example.com/main.cgi?home=main.cgi`

Bileşen adı `main.cgi` Uygulamanın kullandığı normal HTML statik dosyaları ile aynı dizinde yer alır. Bazı durumlarda testçinin istekleri özel karakterler kullanarak kodlaması gerekir (böylece `.` Nokta, `%00` null, vb.) Dosya uzatma kontrollerini atlamak veya komut dosyası yürütülmesini önlemek için.

İpucu: Geliştiricilerin her kodlama biçimini beklememesi ve bu nedenle yalnızca temel kodlanmış içerik için geçerlilik yapması yaygın bir hatadır. İlk başta test ipi başarılı değilse, başka bir kodlama şeması deneyin.

Her işletim sistemi, yol ayırıcı olarak farklı karakterler kullanır:

- Unix benzeri işletim sistemi:
  - Kök dizini: `/`
  - Dizin ayırıcı: `/`
- Windows işletim sistemi:
  - Kök dizini: `<drive letter>:`
  - Dizin ayırıcı: `\` ya da `/`
- Klasik macOS:
  - Kök dizini: `<drive letter>:`
  - Dizin ayırıcı: `:`

Aşağıdaki karakter kodlama mekanizmalarını hesaba katmalıyız:

- URL kodlama ve çift URL kodlama
  - `%2e%2e%2f` temsili `../`
  - `%2e%2e/` temsili `../`
  - `..%2f` temsili `../`
  - `%2e%2e%5c` temsili `..\`

    - `%2e%2e\` temsili `..\`
    - `..%5c` temsili `..\`
    - `%252e%252e%255c` temsili `..\`
    - `..%255c` temsili `..\` Ve böylece.
- Unicode / UTF-8 Kodlama (sadece aşırı UTF-8 dizilerini kabul edebilen sistemlerde çalışır)
    - `..%c0%af` temsili `../`
    - `..%c1%9c` temsili `..\`

Başka işletim sistemi ve uygulama çerçevesine özel hususlar da vardır. Örneğin, Windows dosya yollarının ayrıştırılmasında esnektir.

- Windows kabuğu: Bir kabuk komutunda kullanılan yollardan herhangi birini, işlev farkında olmadan sonuçlanan yollara eklemek:
    - Açılı parantez `<` ve `>` Yolun sonunda
    - Yolun sonunda çifte alıntılar (düzgün bir şekilde kapatıldı)
    - Gibi yabancı akım dizin işaretleri `./` ya da `.\\`
    - Var olabilecek veya olmayabilecek keyfi öğelere sahip yabancı ebeveyn dizini işaretleyicileri:
        - `file.txt`
        - `file.txt...`
        - `file.txt<spaces>`
        - `file.txt""""`
        - `file.txt<<<>>><`
        - `./././file.txt`
        - `nonexistant/../file.txt`
- Windows API: Aşağıdaki öğeler, bir dizenin dosya adı olarak alındığı herhangi bir kabuk komutunda veya API aramasında kullanıldığında atılır:
    - Dönemler

    - Kategori: Mekanlar
- Windows UNC Dosya Yolları: SMB paylaşımlarında referans dosyaları kullanılır. Bazen, uzak bir UNC dosya yolundaki dosyalara atıfta bulunmak için bir uygulama yapılabilir. Eğer öyleyse, Windows SMB sunucusu saldırgana yakalanabilir ve çatlatılabilecek depolanabilir. Bunlar ayrıca filtrelerden kaçınmak için öz referanslı bir IP adresi veya alan adı ile kullanılabilir veya SMSB hisselerindeki dosyalara saldırgana erişilemez, ancak web sunucusundan erişilebilir olarak
  kullanılabilir.
    - `\\server_or_ip\path\to\file.abc`
    - `\\?\server_or_ip\path\to\file.abc`
- Windows NT Cihaz İsim Alanı: Windows aygıtının adının ayırtılması için kullanılır. Bazı referanslar, farklı bir yol kullanarak dosya sistemlerine erişime izin verecektir.
    - Bu gibi bir sürücü mektubuna eşdeğer olabilir `c:\` Hatta atanan bir mektup olmadan bir sürüş hacmi: `\\.\GLOBALROOT\Device\HarddiskVolume1\`
    - Makinedeki ilk disk sürücüsüne atıfta bulunur: `\\.\CdRom0\`

Gray-Box Testing (Gri-Kutu Testi)

Analiz gri kutu testi yaklaşımıyla gerçekleştirildiğinde, testçiler siyah kutu testlerinde olduğu gibi aynı metodolojiyi takip etmek zorundadır. Ancak kaynak kodunu inceleyebildikleri için giriş vektörlerini daha kolay ve doğru bir şekilde aramak mümkündür. Bir kaynak kodu incelemesi sırasında, uygulama kodu içinde bir veya daha fazla ortak kalıp aramak için basit araçları (*kırılgan* komut gibi) kullanabilirler: dahil etme fonksiyonları / meth'ler, dosya sistemi işlemleri vb.

- `PHP: include(), include_once(), require(), require_once(), fopen(), readfile(), ...`
- `JSP/Servlet: java.io.File(), java.io.FileReader(), ...`
- `ASP: include file, include virtual, ...`

Çevrimiçi kod arama motorlarını (örneğin, SearchcodeSearchcode) kullanarak, İnternet'te yayınlanan Open Source yazılımında yol geçişi kusurlarını bulmak da mümkün olabilir.

PHP için, testçiler aşağıdaki reçeteyi kullanabilir:

```
(include|require)(_once)?\s*['"(]?\s*\$_(GET|POST|COOKIE)
```

Gri kutu test yöntemini kullanarak, genellikle keşfedilmesi daha zor olan veya standart bir kara kutu değerlendirmesi sırasında bulunması imkansız olan güvenlik açıklarını keşfetmek mümkündür.

Bazı web uygulamaları, bir veritabanında depolanan değerleri ve parametreleri kullanarak dinamik sayfalar oluşturur. Uygulama veritabanına veri eklediğinde özel olarak hazırlanmış yol traversal
dizeleri eklemek mümkün olabilir. Bu tür bir güvenlik sorununun keşfedilmesi zordur, çünkü kapsayıcılık işlevlerinin içindeki parametrelerin iç ve **güvenli** görünmesi, ancak gerçekte olmadığı gerçeğidir.

Ek olarak, kaynak kodunu inceleyerek, geçersiz girdiyi ele alması gereken işlevleri analiz etmek mümkündür: bazı geliştiriciler, uyarılardan ve hatalardan kaçınmak için geçersiz kılma girişlerini
değiştirmeye çalışır. Bu işlevler genellikle güvenlik kusurlarına eğilimlidir.

Bu talimatlar içeren bir web uygulaması düşünün:

```
filename = Request.QueryString("file");
Replace(filename, "/","\");
Replace(filename, "..\","");
```

Kusur için test yapılır:

```
file=..../....//boot.ini
file=....\\....\\boot.ini
file= ..\..\boot.ini
```

Tools (Araçlar)

- DotDotPwn - The Directory Traversal Fuzzer
- Path Traversal Fuzz Strings (from WFuzz Tool)
- OWASP ZAP
- Burp Suite
- Enconding/Decoding tools
- String searcher “grep”
- DirBuster

References (Referanslar)

Whitepapers (Beyaz Kağıtlar)

- phpBB Attachment Mod Directory Traversal HTTP POST Injection
- Windows File Pseudonyms: Pwnage and Poetry