

**KOCAELİ ÜNİVERSİTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ**

**YAZILIM LABARATUVARI 2**

**3. PROJE**

140201082 SÜMEYRA ÇAKMAK

140201083 MELEK PAZARBAŞI

**ZAYIF WEB UYGULAMASI**

1.Melek Pazarbaşı 2.Sümeyra Çakmak

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

melekp30@gmail.com sumeyra.cak@gmail.com

1.1 PROBLEM TANIMI

"Vulnerable" yani, korunmasız, zayıf olan bir kelimedir. Sistemdeki açıklara veya zayıflıklara denir. Security Hole olarakta geçer ama security hole genelde bir ayarın yapılmamasından doğan açıklardır. Bu açıklar ise genelde bir bugdan doğan açıklardır. Örnek vermek gerekirse, filtrelenmemiş html tagları bir php sayfasında vulnerability olabilir.

Kasıtlı olarak bırakılmış güvenlik zafiyetlerini (vulnerability) içeren zayıf (vulnerable) web uygulamaları, uygulamalı siber güvenlik eğitimlerinde önemli bir rol oynamaktadır. Yasal olmayan bir şekilde gerçek sistemlere saldırmadan bu tip uygulamalar üzerinde öğrendiklerini tecrübe ederek kendilerini geliştirebilmek amacıyla yapılmıştır.

Özellikle bırakılmış olan güvenlik zafiyetlerini içeren bu uygulamada zafiyetlerin sömürülmesi için uygun olan exploitlerin oluşturulması ve bu zafiyetlerin etkisiz hale nasıl getirileceği ile kod içerisinde filtreleme yapılarak uygulamanın güvenliğinin artırılması, Zafiyetin hangi sorundan kaynaklandığı uygun çözümler getirerek gerekli işlemlerin yapılması.

**1.2 YAPILAN ARAŞTIRMALAR**

Web uygulaması üzerinde içermesi gereken zafiyetlerle ilgili araştırmalar yapılmıştır;

1. SQL Enjeksiyonu(SQL Injection)
   1. SQL nedir
   2. SQL Injection
      1. SQL Injection Nasıl Yapılır?

1.2.2 SQL ınjectiondan Nasıl Korunulur?

* + 1. Meta-Karakter

1. XSS- Cross Site Scripting(Siteler Arası Betik Çalıştırma)

2.1 Zafiyet Neden Kaynaklanmakta

2.2 Saldırı Türleri

2.2.1 Stored(Persistent)

Xss Saldırısı

2.2.2 Reflected Xss Saldırısı

2.2.3 DOM Xss Saldırısı

2.3 Xss Zafiyetinin Çözümü

1. Command Injection

(Komut Enjeksiyonu)

3.1. *Command Injection Saldırısı Nasıl Yapılır?*

*1-) SQL INJECTION* :

## *1.1 SQL Nedir?*

**SQL** (*Structured Query Language*) veritabanlarında data çekme, silme ve değiştirme gibi işlemler için kullanılan basit yapılı bir dildir. Bugün hemen hemen tüm web uygulamalarının altyapısında veritabanı desteği vardır ve bu web uygulamaları veritabanı ile SQL aracılığıyla anlaşırlar.

Bir siteye mesaj bıraktığınızda bu mesaj veritabanına kaydedilir. O mesaj onaylandığında veritabanındaki bir alan güncellenmiş olur. Yönetici veritabanındaki kaydı silerek o mesajın siteden silinmesini sağlar.Örnek bir kayıt silme SQL cümlesi şu şekilde olabilir;

---DELETE FROM members WHERE id=17

Yukarıdaki kod veritabanı tarafından çalıştırıldığında members tablosunda id alanı 17 olan kayıt silinecektir. [1]

## *1.2 SQL Injection*

Web uygulamalarında birçok işlem için kullanıcıdan alınan veri ile dinamik SQL cümlecikleri oluşturulur. Mesela “SELECT \* FROM Products” örnek SQL cümleciği basit şekilde veritabanından web uygulamasına tüm ürünleri döndürecektir.Bu SQL cümlecikleri oluşturulurken araya sıkıştırılan herhangi bir meta-karakter SQL Injection’a neden olabilir.

SQL Injection, SQL sorgusunun amacına müdahale ederek farklı bilgileri elde etmeye denir.

Yani SQL Injection yöntemi ile üye bilgileri, yönetici şifreleri gibi veritabanında bulunup herkese açık olmayan bilgiler elde edilebilir.

Sql Injection veritabanından ve dilden bağımsız olarak her türlü uygulama veritabanı ilişkisine sahip sistemde bulunabilir ve veritabanının bir açıpı değildir. Sql Injection’dan korunmak web geliştiricisinin görevidir. [1]

*1.2.1 SQL Injection Nasıl Yapılır?*

Diyelim SQL kullanan bir siteye user 123 kullanıcı adı ve şifresiyle giriş yapıyorsunuz. Giriş butonuna tıkladığınız anda sitenin yazılımı veritabanına aşağıdaki gibi bir sorgu gönderecektir:

SELECT \* FROM kullanicilar WHERE isim='user' AND sifre='123'

Yazılım sonuç bulunursa girişinize izin verecek, bulunamazsa izin vermeyecektir. Bir de kullanıcı adı ve şifre alanına ' OR 1=1 -- yazdığımızda gönderilecek olan sorguya bakalım:

SELECT \* FROM kullanicilar WHERE isim=''OR 1=1 --' AND sifre=''OR 1=1 --'

Yukarıdaki sorgu gönderilecektir. Eğer yazılım girdileri filtreliyorsa SQL injectionu engelleyip girişinize izin vermeyecek, filtrelemiyorsa kullanıcı adı ve şifrenizi doğru zannedip girişe izin verecektir.  
   
' OR 1=1 -- SQL dilinde her zaman true döndürür (olumlu sonuç döndürür). Bu yüzden SQL programı bütün kayıtları listeleyeceğinden, yazılım doğru giriş yapıldı zannedecektir. "' OR 1=1 --" ifadesi, önceki koşul gerçekleşmese bile boş ver, 1=1 gerçekleşiyorsa true döndür ve sorgunun geri kalanını boş ver (-- yorum anlamına geldiğinden sorgunun kalanı yorummuş gibi algılanır) demektir.  
   
Bir örneği daha inceleyelim. SQL injectiona karşı önlem alınmamış bir makale sitesi olsun. Makaleler adres çubuğunda yazılan id'e göre gösterilsin.

makale.php?id=1

SQL açığı olduğunu bilmediğimizi varsayarak 1 yerine ' yazalım. SQL açığı olduğu için SQL hatası verecektir. Artık bu açığı kullanarak veritabanından istediğimiz veriyi elde edebiliriz. Zaten makale gösterilirken bir SELECT kullanıldığı için, başka verileri almak için UNION SELECT kullanıyorum.

makale.php?id=1' UNION SELECT \* FROM users

Sayfada bütün kullanıcılar ve şifreleri listelenecektir. [2]

*1.2.2 SQL Injectiondan Nasıl Korunulur?*

Kullanıcıdan alınan her veride ' ve " karakterleri getirilerek korunulur. Bu karakterler ya karakter koduna çevrilmeli, ya da başına \ yani kaçırma karakteri eklenmelidir. Bu durumda SQL programı o karakterleri komut olarak algılamayacaktır.[2]

*1.2.3 Meta-Karakter*

Meta-karakter bir program için özel anlamı olan karakterlere verilen isimdir. Örnek olarak C temelli C#, Javascript, PHP gibi dillerde (\) backslash karakteri bir meta-karakterdir. Compiler (derleyici) ya da Interpreter (yorumlayıcı) bu karakteri görünce ondan sonraki karakteri ona göre işler.

SQL’ için kritik metakarakter (‘) tek tırnak’ tır. Çünkü iki tek tırnağın arası string olarak algılanır. Diğer bir önemli meta-karakter ise (;) noktalı virgüldür, satırın bittiğini ve yeni satır başladığını bildirir.

Genel bir web uygulamasında olası bir üye girişi işlemi şu şekildedir;

* Formdan gelen kullanıcı adı ve şifre bilgisi ile ilgili SQL cümleciği oluşturulur (SELECT \* FROM members WHERE user=’admin’ AND password=’sifre’ gibi)
* SQL cümleciği kayıt döndürüyorsa böyle bir kullanıcının var olduğu anlamına gelir ve session(oturum) açılır ve ilgili kullanıcı üye girişi yapmış olur.
* Eğer veritabanından kayıt dönmediyse "kullanıcı bulunamadı" veya "şifre yanlış" gibi bir hata ile ziyaretçi tekrar üye girişi formuna gönderilir. [1]

*2-) XSS-CROSS SITE SCRIPTING:*

*(Siteler Arası Betik Çalıştırma)*

XSS (Siteler arası betik çalıştırma) zafiyeti, saldırganın html, css, javascript ile hazırlamış olduğu zararlı kod parçalarının hedef kullanıcının(kurbanın) browserında izinsiz olarak çalıştırmasına imkan tanıyan bir web uygulama güvenliği zafiyetidir. Bu açık türü de nerdeyse tüm web application açık türleri gibi, input validation eksikliğinden kaynaklanır. Yani dışarıdan kullanıcının girebildiği parametrelerin filtrelenmemesinden kaynaklanır. Bildiğiniz gibi sql injection da bu sebeple ortaya çıkmaktadır. Ama bu saldırı türünde sayfada "html" veya "javascript" kodları çalıştırmak mümkündür.

 Başka bir deyişle; bir uygulamada bulunan XSS zafiyeti saldırgana, hedef kullanıcının tarayıcısında zararlı kod çalıştırma imkanı tanır. Bu imkan neticesinde saldırgan, hedef kullanıcının oturum bilgilerini, ekran görüntüsünü, tuş girişleri gibi bilgileri alabilir, uygulama içeriğinin manüpüle edebilir. Bu zafiyet istismar edilirken bazen kurbanın insiyatifine bağlı olurken(Reflected ve DOM based türlerinde) bazen de saldırgan, kurban ile muhattap olmadan da zafiyetten etkilenmesini sağlayabilir(persistent türünde). [3]

*2.1 Zafiyet Neden Kaynaklanmakta?*

XSS saldırıların en temel nedeni kullanıcılardan alınan inputların hiçbir filtrelemeden geçmeden işleme tabi tutulmasıdır.

Bu inputlar;

* kullanıcıdan form elemanları aracılığıyla alınan bir değer olabilir(search, login, register etc.),
* get metoduyla gönderilen bir değer olabilir,
* http headerlerı ile gönderilen bir değer olabilir,
* cookie, session id değerleri olabilir,
* bir file upload kısmında dosyanın kendisi veya dosyanın adı olabilir.[3]

*2.2 Saldırı Türleri*

XSS saldırısında amaç; hedef kullanıcının tarayıcısında bir şekilde zararlı kod çalışmaktır. Bu amaca ulaşmak için bir kaç farklı yöntem bulunmaktadır. Bu farklılıktan dolayı xss saldırıları şimdilik 3 türe ayrılmıştır.

*2.2.1 Reflected Xss Saldırısı*

Reflected XSS saldırısında; kurbanın, hedef siteye istek yapması için kullanacağı bağlantıda(link) zararlı kod parçası bulundurmasıdır. İstek yapılırken bu zararlı kod ifa edilir ve dönen cevap saldırganın saldırganın kontrolunde olan bir uzak sunucuya gönderilir. Burada önemli olan nokta; zafiyetin istismar edilmesi tamamen kullanıcının insiyatifine kalan bir durumdur. Yani kullanıcı zararlı kod içeren bağlantıya tıklamadığı sürece zafiyetten etkilenmeyecektir. Ayrıca diğer türlerine oranla en çok karşılaşılan xss saldırı türüdür.[3]

*2.2.2 Stored (Persistent) Xss Saldırısı*

Kullanıcıdan alınan verinin yeterli filtrelemeden geçmemesi sonucunda veri tabanına kayıt edildikten sonra kayıt edilen bu veri başka bir yerde kullanılmak üzere veri tabanından çekileceği sırada ortaya çıkan bir xss zafiyet türüdür. Diğer türlerine oranla çok daha tehlikelidir. Çünkü bu xss zafiyet türünde zararlı kod veri tabanına kayıt edilir. Bu da şu anlama gelmektedir; Sisteme kayıtlı olan kullanıcılar zafiyetten etkilenen sayfayı ziyaret ettikleri anda oturum bilgilerini farkında olmadan saldırgana kaptırırlar. Tehlikeli olan nokta tam da burası işte. Saldırganın kimseyle muhattap olmaması... Diğer xss türlerinde saldırgan, kullanıcılara ilgili bağlantıya bir şekilde istek yaptırtmaya çalışır ama burada böyle bir durum söz konusu değildir. Payloadın kendisi sitenin veri tabanında kayıtlı zaten. Sadece payloadın select edileceği sayfayı kullanıcın ziyaret etmesi yeterli. Bu durumda sadece bir kişi veya bir grup değil sistemde kayıtlı olan herkes zafiyetten etkilenmiş olur. Diğer xss türlerinde fazla detaya girdiğimiz için bu türün teknik olarak diğerlerinden çok bir farkı bulunmamakta. Sadece bu sefer işin içinde veri tabanı girmektedir. Bu da saldırının kapsamını ve tehlikesini ciddi anlamda büyütmektedir.[3]

*2.2.3 DOM Xss Saldırısı*

 Yapısı gereği diğer iki türden farklıdır. Çünkü persistent ve reflected xss zafiyetleri sunucu taraflı filtrelemeler ile engellenebilirken DOM tabanlı xss de böyle bir durum söz konusu değildir.

DOM XSS saldırısı en tehlikeli XSS saldırı türüdür. JavaScript kodlarının zararlı bir şekilde kullanılması ile yapılır.  
Bu saldırı türünde kullanılan DOM nesneleri ile hedef web sitesinin index’i değiştirilebilir, sayfanın kodları ile oynanabilir, virüs, trojan gibi zararlı kodlar sayfaya entegre edilebilir.[3]

*2.3 XSS Zafiyetinin Çözümü*

#### Girdi Kontrolleri

Zafiyete, kullanıcının müdahale edebildiği alanlar veya kullanıcıdan alınan veriler sebep olduğu için çözümü de burada arayacağız. Girdi denetimleri çok sıkı sıkıya yapıldığı takdirde bu zafiyet ortaya çıkmayacaktır. Bu denetimlerde kullanılan çözüm teknikleri.

#### White List Tekniği

Pozitif girdi denetimi olarakta bilinen bu çözüm metodunda kullanıcıdan gelecek olan verilerin(karakterlerin, kelimelerin vs) hangilerine izin verileceği belirtilir. Örnek vermek gerekirse kullanıcıdan sadece alfanumeric değerler alıyor isek (yani A'dan Z'ye ve 0'dan 9'a) bunu regex ile ifade ederek sadece kabul edeceğimiz verileri belirleriz. Bu durumda kullanıcıdan gelecek olan özel karakterlerin bütününü kabul etmemiş oluruz. Başka bir örnek daha verecek olursak kullanıcıdan aldığımız değerler sadece belli kelimeler veya ifadelerden ibaret ise sadece kabul edebileceklerimizi belirler, belirlediğimiz değerler dışında gelen değerleri işleme almayız.

#### Black List Tekniği

Bu teknikte ise white list'in aksine kullanıcıdan gelen veriler arasında kabul etmediklerimizi belirleriz. White list gibi sağlam görünse de aslında hiç öyle değildir. Bu çözüm tekniğinde olasılıklar çok fazla ve bir tanesinin bile gözünüzden kaçması zafiyete sebeb olmaktadır. < , ' , > , " karakterlerini engellediğinizde bunların hex formatını da engelleyeceksiniz encode edilmiş hallini de engelleyeceksiniz yazılan koda göre değişiklik göstermekle beraber bazen bu karakterler kullanılmadan da zafiyet oluşabilmektedir. Bu durumda script kelimesini engellemelisiniz, alert, prompt, confirm, hex formatları, char formatları vs vs külfetten başka bir şey değil gördüğünüz gibi olasılıklar çok fazla çünkü.

*3-) COMMAND INJECTION* :

Command Injection, yani komut enjeksiyonu saldırganın zafiyet barındıran bir uygulama üzerinden hedef sistemde dilediği komutları çalıştırabilmesine denir. Komut ile kastedilen şey Windows'ta CMD ve Linux'ta Terminal pencerelerine girilen sistem komutlarıdır. Literatürde Shell kodlaması diye de geçer. Command Injection saldırısı büyük oranda yetersiz input denetleme mekanizması nedeniyle gerçekleşmektedir.[4]

*3.1 Comman Injection Saldırısı Nasıl Yapılır?*

Metin kutusuna gireceğiniz bir IP adresi ya da bir domain adresi sonucu ekrana 4 tane ping paketinin belirtilen adrese gönderildiğini göreceksiniz.

Buradan şu tespite varmanız gerekir: Eğer ping paketleri gönderiliyor ve alınıyorsa - ki çıktı bunu söylüyor - bu durumda web uygulaması sunucu tarafında CMD ya da Terminal komutları çalıştırıyor demektir. Bu tespitten sonra ikinci tespit etmeniz gereken nokta ise metin kutusuna girdiğiniz IP ya da domain adresinin sunucu tarafındaki CMD/Terminal komutlarına eklendiği ve bu şekilde CMD/Terminal komutlarının çalıştırıldığıdır. Çünkü çıktıya dikkat edecek olursanız metin kutusuna girilen adresin aynısı çıktıda da mevcuttur. Yani ping paketleri metin kutusundaki adrese gönderilmiştir.

Bu tür bir hizmet veren web uygulamasında eğer metin kutusu denetlemeye tabi tutulmamışsa saldırgan komut satırı kodlama bilgisini kullanarak var olan ping komutunun yanına kendi komutunu ekleyebilir. Böylelikle ping komutunu çalıştıran sunucu istemeden saldırganın gönderdiği komutu da çalıştıracaktır ve ekrana yansıtacağı çıktının içinde saldırganın eklediği komutun çıktısı da yer alacaktır.

|  |
| --- |
| echo shell\_exec('ping -c 4 ' . $metinKutusu); |

shell\_exec() fonksiyonu, içerisinde yer alan ping komutunu sunucunun komut satırı üzerinde çalıştıracaktır ve echo komutuyla da çalıştırılan komutun çıktısı ekrana yansıtılacaktır. Yukarıdaki satırda yer alan -c 4 ifadesi 4 kere ping paketi gönder anlamına gelir ve $metinKutusu değişkeni ise ekrandaki metin kutusuna girilen veriyi tutan bir değişkeni temsil eder. Şimdi bu PHP kodunu gördükten sonra command injection'ı daha iyi idrak edebiliriz. $metinKutusu'na girilecek bazı operatörlerle yeni komutlar dahil edilebilmektedir. Eğer shell (komut satırı) diline biraz vakıfsanız bilirsiniz ki &&, | ya da || gibi operatörler, ayrıca ; gibi sonlandırıcılar shell komutlarını birbirlerinden ayıran ya da birbirlerine bağlayan özelliğe sahiptirler. İşte bu operatörler yardımıyla ping komutunun sonuna yeni bir komut ekleyeceğiz. Böylece command injection teşebbüsünde bulunmuş olacağız.

*Sayfa Hack'lemek*

Şimdi ilgili zafiyetin bulunduğu sayfayı

hack'leyelim ve sayfada sadece "Hacked By Script Kiddies" yazısının görüntülemesini sağlayalım.

|  |
| --- |
| www.includekarabuk.com && echo "<font color=red><center><h1>Hacked By Script Kiddies</h1></center></font><br>" > index.php |

Yukarıdaki enjeksiyon kodundaki echo komutu argüman olarak aldığı string'i ekrana basmaya yarar. > operatörü ise kendinden önceki komutun output'unu kendinden sonraki dosyanın içeriğine yazmaya yarar. [4]

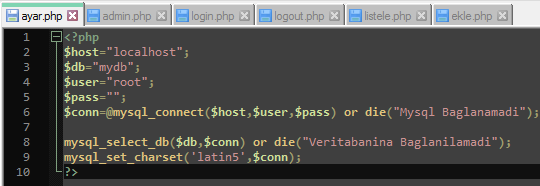
**1.3 TASARIM**

***1.3.1 VERİTABANI DİYAGRAMI:***

uyeler

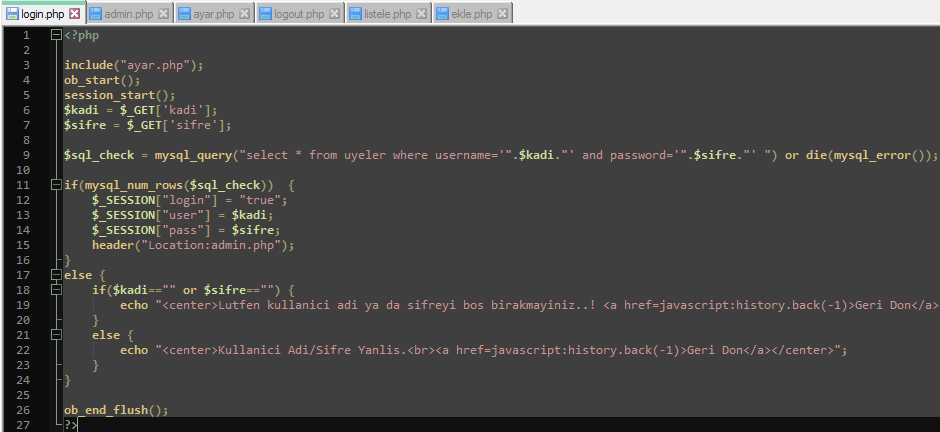
*Şekil 1* :Veritabanı ER diyagramı

**1.3.2 YAZILIM MİMARİSİ:**



*Şekil* 2 : Sunucu ile veritabanı arasındaki bağlantı (ayar.php)

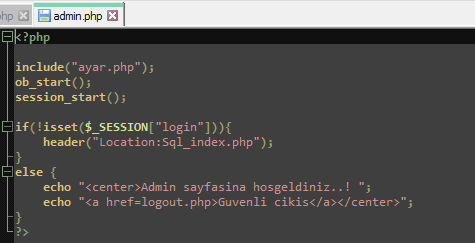
Veritabanındaki kolonlara uygun olarak veritabanı bağlantısı yapılmıştır.



*Şekil* 3: login.php

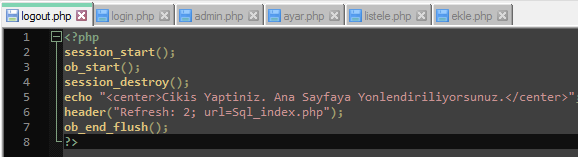
Sql injection için çalışıyor.

Login sayfasında girilen kullanıcı adı ve şifre değerleri veri tabanındaki değerlerle uyum sağlarsa admin.php sayfasına yönledirir.Giriş başarılı olmuş olur.



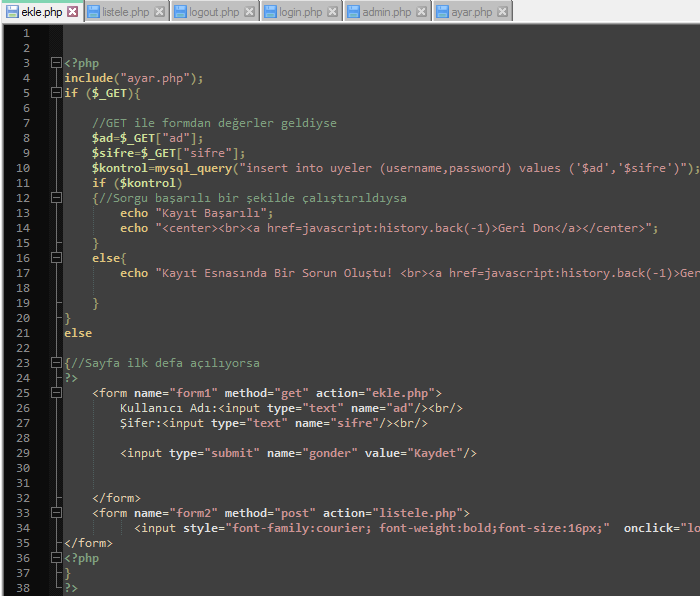
*Şekil* 4: admin.php

Bu sayfaya erişim sağlandığında giriş başarılı olmuş olur. Eğer giriş başarılı olmasa tekrardan login.php sayfasına dönüş yapılır.



*Şekil* 5 : logout.php

Çıkış yapmak istediğimiz zaman login.php sayfasına yönlendiriliyoruz.



*Şekil* 6: ekle.php

Xss cross site sctripting zafiyeti için çalışıyor.

Ekle.php sayfasında öncelikle olarak form sayfasında gerekli alanları doldurduktan sonra veritabanına kayıt yapılır.Bu girilen stringler html tagları arasına yazıldığı takdirde de veri tabanına html tagları ile yazılır.



*Şekil* 7 : listele.php

Ekle.php sayfasında girilen değerleri veritabanına atar ve daha sonra listele butonuna basıldığında veri tabanındaki verileri listeler.Html tagları kullanılarak kaydedilmiş verileride listelerken taglara uygun olarak listeler.Çünkü kod içerisinde zafiyet bulunmaktadır.

* 1. **GENEL YAPI:**

**1.4 GENEL YAPI:**

# Projede vulnerable web uygulaması yapılması istendi. Buna uygun olarak bir web sitesi tasarımı oluşturuldu.Zafiyetler veri tabanı ile ilişkilendirilerek sunucu tarafında oluşturulmuştur.Oluşturulan zafiyetler kullanıcının girdilerinde değişiklik yapmaya yöneliktir.Projenin veri tabanı bağlantısını sağlamak için ayar.php(*Şekil* 2) kodu üzerinden bağlantı sağlanarak sql injection zafiyet uygulaması için login.php(*Şekil* 3) sayfasına giderek ilgili alanları zafiyet url’si yardımıyla doldurarak ilgili zafiyeti sömürebilir.Daha sonrasında admin.php(*Şekil 4*) sayfasından çıkış yapmak istediği zaman logout.php (*Şekil 5*)sayfasına yönlenir.

Xss Cross site scripting zafiyeti için

ekle.php(***Şekil 6***) sayfasındaki ilgili alanların html taglarıyla veya normal stringler yardımıyla doldurulmasıyla kayıt butonuna basıldığında veri tabanına kaydedilen veriler listele butonuna basılmasıyla listele.php (***Şekil7*** ) sayfasına yönlenerek veri tabanındaki tüm verileri php sayfasında gösterir ve eğer kayıt esnasında girdiğimiz stringleri html taglarıyla girişini yaptıysak o şekilde veri tabanına kaydedip listeleme kısmında hiçbir filtreleme yapılmadığı için html taglarına uygun olarak işleterek karşımıza çıkarır.

# 1.5 REFERANSLAR

1. <http://ferruh.mavituna.com/sql-injection-a-giris-ve-sql-injection-nedir-oku/https://www.youtube.com/watch?v=Jc5eXYwTROg>
2. <http://www.kodevreni.com/526-sql-injection-nedir-ve-sql-injection-nas%C4%B1l-%C3%B6nlenir/https://cmengcompany.wordpress.com/2011/06/21/c-thread-kullanimi/>
3. <http://musana.net/xss-detayli-dokuman>.
4. <http://www.includekarabuk.com/kategoriler/DVWAUygulamasi/Ders-4---Command-Injection-Low-Level.phphttp://mail.baskent.edu.tr/~20410964/DM_8.pdf>
5. <https://www.phpr.org/php-mysql-baglantisi-yapmak/>
6. <https://www.mkoseoglu.com/php-mysql-veritabanina-bilgi-ekleme>
7. <https://www.msxlabs.org/forum/bilgisayar/210796-php-ile-veri-tabanindaki-bilgileri-listelemek.html>