# YAZILIM LABORATUVARI PROJE 3

# ZAYIF WEB UYGULAMALARI GELİŞTİRİLMESİ PROJE DÖKÜMANI

**1-Kurulum Yapılabileceği İşletim Sistemleri**

Windows 7 ve sonrasında,Linux/GNU,Mac OS X ...

**2- Web sunucu yazılımının kurulumu**

Localhost için Tomcat Server kullanıldı ve bu Intellij IDE'sin de kullanıldı.

Öncelikle bu url den Tomcat'i indirelim http://tomcat.apache.org/

Daha sonra indirdiğimiz dosyayı intellij'de ayarlar kısmına girerek 'Application Servers' a tıklıyoruz ve + simgesine tıklayarak Tomcat'i ekliyoruz.

Daha sonra Run sekmesinden Edit Configuration'a giriyoruz ve karşımıza Run/Debug Configurations çıkıyor '+' kısmına tıklıyoruz ve buradan Tomcat Server'ı bularak 'local' e tıklıyoruz .

Deployment sekmesine gelerek tekrar '+' simgesine tıklıyoruz ve [projemizin adı]:war exploded yazan deployment'a tıklıyoruz bu bizim server startup'ımız oluyor.Bunu yaptıktan sonra en altta 'before launch' adında bir kısım orada da '+' simgesine tıklıyoruz ve Build Artifact diyoruz yukarda seçtiğimiz war exploded 'ı seçiyoruz.bu şekil de işlemimizi tamamlıyoruz artık serverımızı çalıştırabiliriz.

İsteğe bağlı Maven build'i de eklenebilir.

**3-Veri Tabanı Kurulumu**

Veri tabanı olarak mysql kullanıldı.Ubuntu için kurulumu.Server kurulumunu Windows'ta yaptık 'Bash on Ubuntu on Windows' ta ise mysql kurulumunu gerçekleştirdik.

https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/

Yukarıdaki linkten Bash Ubuntu kurulumu yapılabilir.

Öncelikle hostname kontrolü yapılıyor.

hostname

hostname -f

Daha sonra sitem güncelleştirilmesi yapılıyor.

sudo apt-get update

sudo apt-get upgrade

Yükleme için komut

sudo apt-get install mysql-server

Daha temiz ve daha güvenli mysql için

sudo mysql\_secure\_installation

MySql'e root kullanıcı olarak giriş yapalım

mysql -u root -p

Eğer yukarıdaki gibi secure girildiyse password soracaktır onu girin.Ve mysql'e adımımızı atmış oluyoruz.

**4-Veri Tabanın Kurulumu**

Mysql 'e yukarıda giriş yapmıştık.Şimdi bir database oluşturup kullanıcı oluşturma ile veritabanının kurulumunu gerçekleştirilicek.

mysql database ve kullanıcı oluşturma

**-------------------------------------------------------------------------------------**

**create database koudb**

**CREATE USER 'kou'@'localhost' IDENTIFIED BY 'kou123' ;**

**CREATE USER 'kou'@'%' IDENTIFIED BY 'kou123' ;**

**GRANT ALL ON \*.\* TO 'kou'@'localhost' ;**

**GRANT ALL ON \*.\* TO 'kou'@'%' ;**

**FLUSH PRIVILEGES ;**

**-------------------------------------------------------------------------------------**

Yukarıda görüldüğü üzere kurulumu tamamladık.

**1.1-SQL Injection Zafiyeti**

İnternet sitelerinin bir çoğunda, sayfayı dinamik tutmak için veritabanından yararlanılır. Güncel veritabanı yazılımlarında birçoğunda (MySQL, MSSQL,  Sqlite, Oracle SQL) SQL (Structured Query Language) denilen ortak bir dil kullanılır.  
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 kullanılarak Yahoo! gibi büyük internet sitelerinin bile veritabanları ele geçirilmiştir.

**1.2-Güvensiz Kodlama**

"select \* from user where username='" + username + "' and password='" + password + "'";

Kodda kullanılan bu SQL komutunda user tablosundan x kişisinin adına ve x kişisinin şifresine sahip olan veriyi getirmektedir.İşte burada bulunan zafiyet login sayfasın da gelen username ve password'un kontrollerinin zayıf olmasıyla ilgilidir.

Session currentSession = sessionFactory.getCurrentSession();  
String sql = "select \* from user where username='" + username + "' and password='" + password + "'";//SQL string  
  
SQLQuery query = currentSession.createSQLQuery(sql);//SQL komutunun querye eklenmesi  
query.addEntity(User.class);

**1.3-Sömürü**

"select \* from user where username=' ' or 1=1 ' AND password=' ' or 1=1';

girildiğinde yani username ve passworde yukarıdaki karakterler girildiğinde SQL dönen komut ile ilk veri tabanından gelen herhangi bir kullanıcı ile sisteme giriş yapılmış olacaktır.Bu şekilde sistemdeki kişilerin bilgileri sömürülebilir.

Örneğin aşağıdaki get isteği yapıldığında tablodaki ilk kullanıcı ile giriş yapılacaktır.

http://localhost:8080/sqlInjection?username=' or '1'='1&password=' or '1'='1

**1.4-İyileştirme**

Eğer aldığımız değerleri direkt sql query’de kullanmak yerine özel karakterleri filtreleyebiliriz. Bizim örneğimizde where kısmının Criteria ile yaptığımız durumda filtreleme otomatik yapılacaktır.

**Criteria criteria = sessionFactory.getCurrentSession().createCriteria(User.class);**

criteria.add(Restrictions.*eq*("username", username));

criteria.add(Restrictions.*eq*("password", password));

return criteria.list().size() > 0 ? (User) criteria.list().get(0) : null;

**2.1-XSS-Cross Site Script Zafiyeti**

XSS (Cross Site Scripting) script kodları üzerinden (genelde javascript) bir web sayfasına saldırı yapılmasıdır.XSS çoğunlukla tarayıcıda saklanan bilgiler olan cookielere saldırı amacı ile kullanılmaktadır.

**2.2-Güvensiz Kodlama**

<**c:forEach** var="comment" items="**${**comments**}**">

<li><**c:out** escapeXml="false" value="**${**comment.message**}**"></**c:out**></li> // gelen deger xml’den kacmadan yaziliyor

</**c:forEach**>

XSS script kodları üzerinden gerçekleştirilen bir saldırılır ve bu zafiyeti gidermek için herhangi bir kontrol konulmadığından dolayı kolaylıkla erişim sağlanır.

**2.3-Sömürü**

Sömürü kolaylıkla' <script>"herhangi bir komut "</script> ' bu şekilde sağlanabilir. Örneğin <script>alert('XSS')</script> şeklinde siteye bir comment eklediğimiz de giren kullanıcılara XSS diye bir alet gösterilecektir.Bu şekilde zafiyetten yararlanılarak bir çok saldırı yapılabilir.

Örneğin aşağıdaki gibi bir istek yapıldığında bu sayfayı görüntüleyen tüm kullanıcılarda Hello yazan bir alert görünecektir.

http://localhost:8080/xss?postComment=%3Cscript%3Ealert(%27hello%27);%3C/script%3E

**2.4-İyileştirme**

Kullanıcının yorum olarak girdiği değeri gösterir iken html karakterini es geçer isek bu saldırıyı engellemiş oluruz.

<**c:forEach** var="comment" items="**${**comments**}**">

<li><**c:out** escapeXml="true" value="**${**comment.message**}**"></**c:out**></li> // bu durumda girilen html veyaz js calismayacaktir

</**c:forEach**>

**3.1-Command Injection Zafiyeti**

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.

**3.2-Güvensiz Kodlama**

public static String executeCommand(String command)//komutun çalıştığı kısım  
{  
 command = "cmd.exe /c " + command;  
 StringBuffer output = new StringBuffer();  
  
 Process p;  
 try  
 {  
 p = Runtime.*getRuntime*().exec(command);  
 p.waitFor();  
 BufferedReader reader = new BufferedReader(new InputStreamReader(p.getInputStream()));  
  
 String line;  
 while ((line = reader.readLine()) != null)  
 {  
 output.append(line + "\n");  
 }  
  
 }  
 catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
  
 return output.toString();  
  
}

Yukarıda komutun gönderildiği ve çalıştırıldığı fonksiyon bulunmaktadır.Aşağıda ise command'in gönderildiği controller bulunmaktadır.Fakat controller'a gelen host için herhangi bir kontrol yapılmamaktadır.Controller içinde host'la ilgili bazı kontroller yapılması gerekmektedir gelen host'a bazı kısıtlamalar getirilmeli yada karakterleri belli bir patterne göre alınmalıdır ki injection engellensin.

if (*isNotBlank*(host))//Basit blank kontrolu  
 {  
 String command = "ping " + host; //gelen hostun command stringine eklenir.  
 model.addAttribute("result", *executeCommand*(command)); // execute olması için fonksiyonuna gönderilir  
 }

**3.3-Sömürü**

Saldırgan cmd'ye rahat bir şekilde ulaşabildiği için artık bir çok şeyi yapabilir.Mesela 1;ls yazdığımız da 1 sayısı ile 1 adresine ping at demiş oluyoruz bu geçersiz bir komut olarak sayılacağı için ekrana sadece gönderdiğimiz komut ile ilgili bilgiler gelecektir.ls komutu ile de dizindeki dosyalarının isimlerini görüyoruz cd .. ile devam ederek dosyalar arasında geçişler yaparak tüm dosyalarına ulaşabiliriz.Bu yönden çok tehlikeli bir zafiyettir.

Örneğin aşağıdaki örnekte olduğu gibi kendi kodumuzu çalıştırabiliriz. Burada son kısımda yazan işlem **echo test** bizim ekledigimiz bir koddur ve calisacaktir.

[http://localhost:8080/commandInjection?host=www.google.com%26%26echo%20test](http://localhost:8080/commandInjection?host=www.google.com%26%26echo test)

**3.4-İyileştirme**

Eğer gelen komutu direkt olarak çalıştırmayız aşağıdaki gibi bir filtreden geçirir isek bu saldırıyı engellemiş oluruz. Bizim örneğimiz için aşağıdaki gibi sadece beklenen url pattern’i varsa sadece o değer için işlem yapılacaktır.

private String filterHost(String command)

{

String regex = "www.[a-z-Z]{0,}.(com|net|org)";

Pattern r = Pattern.*compile*(regex);

Matcher m = r.matcher(command);

if(m.find())

return m.group();

return "";

}