Website Vulnerability Scanner Report

✓ http://testing1.pentest-tools.com/dvwa/

Summary

Overall risk level:

High

Risk ratings:

High:	5
Medium:	3
Low:	3
Info:	8

Scan information:

Start time:	2019-05-24 09:07:56
Finish time:	2019-05-24 09:11:22
Scan duration:	3 min, 26 sec
Tests performed:	19/20
Scan status:	Finished

Findings

Vulnerabilities found for server-side software

Risk Level	CVSS	CVE	Summary	Exploit	Affected software
●	7.5	CVE-2017-7679	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.	N/A	http_server 2.4.25
●	7.5	CVE-2017-7668	The HTTP strict parsing changes added in Apache httpd 2.2.32 and 2.4.24 introduced a bug in token list parsing, which allows ap_find_token() to search past the end of its input string. By maliciously crafting a sequence of request headers, an attacker may be able to cause a segmentation fault, or to force ap_find_token() to return an incorrect value.	N/A	http_server 2.4.25
●	7.5	CVE-2017-3169	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_ssl may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during an HTTP request to an HTTPS port.	N/A	http_server 2.4.25
●	7.5	CVE-2017-3167	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.	N/A	http_server 2.4.25
●	7.2	CVE-2019-0211	In Apache HTTP Server 2.4 releases 2.4.17 to 2.4.38, with MPM event, worker or prefork, code executing in less-privileged child processes or threads (including scripts executed by an in-process scripting interpreter) could execute arbitrary code with the privileges of the parent process (usually root) by manipulating the scoreboard. Non-Unix systems are not affected.	N/A	http_server 2.4.25

Details

Risk description:
These vulnerabilities expose the affected applications to the risk of unauthorized access to confidential data and possibly to denial of service attacks. An attacker could search for an appropriate exploit (or create one himself) for any of these vulnerabilities and use it to attack the system.

Recommendation:
We recommend you to upgrade the affected software to the latest version in order to eliminate the risk of these vulnerabilities.

Cross-Site Scripting

Vulnerable Page	Vulnerable Parameter	Method	Attack Vector	
/dvwa/login.php	username	POST	http://testing1.pentest-tools.com/dvwa/login.php **POST Data:** username=</div><script>alert(1);</script><div>	

/dvwa/vulnerabilities/brute/	username	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/brute/?Login=Login&password=ZAP&username=%27%22%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	🚀
/dvwa/vulnerabilities/sqli/	id	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/sqli/?Submit=Submit&id=%27%22%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E	🚀
/dvwa/vulnerabilities/xss_r/	name	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/xss_r/?name=%3C%2Fpre%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cpre%3E	🚀
/dvwa/vulnerabilities/xss_s/	txtName	POST	http://testing1.pentest-tools.com/dvwa/vulnerabilities/xss_s/ **POST Data:** txtName=</div><script>alert(1);</script><div>	🚀
/dvwa/vulnerabilities/xss_s/	mtxMessage	POST	http://testing1.pentest-tools.com/dvwa/vulnerabilities/xss_s/ **POST Data:** mtxMessage=</div><script>alert(1);</script><div>	🚀

Details

Risk description:
The risk exists that a malicious actor injects JavaScript code and runs it in the context of a user's session in the application. This could potentially lead to various effects such as stealing session cookies, calling application features on behalf of another user, exploiting browser vulnerabilities.

Successful exploitation of Cross-Site Scripting attacks requires human interaction (ex. determine the user access a special link by social engineering).

Recommendation:
There are several ways to mitigate XSS attacks. We recommend to:
- never trust user input
- always encode and escape user input (using a Security Encoding Library)
- use the HTTPOnly cookie flag to protect from cookie theft
- implement Content Security Policy
- use the X-XSS-Protection Response Header

References:
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

SQL Injection

Vulnerable Page	Vulnerable Parameter	Method	Attack Vector	
/dvwa/vulnerabilities/brute/	username	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/brute/?Login=Login&password=ZAP&username=ZAP	🚀
/dvwa/vulnerabilities/sqli/	id	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/sqli/?Submit=Submit&id=ZAP%27+AND+%271%27%3D%271%27+--+	🚀
/dvwa/vulnerabilities/sqli_blind/	id	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/sqli_blind/?Submit=Submit&id=ZAP%27+AND+%271%27%3D%271%27+--+	🚀
/dvwa/vulnerabilities/xss_s/	btnSign	POST	http://testing1.pentest-tools.com/dvwa/vulnerabilities/xss_s/ **POST Data:** btnSign=Sign Guestbook" AND "1"="1" --	🚀

Details

Risk description:
The risk exists that an attacker gains unauthorized access to the information from the database of the application. He could extract information such as: application usernames, passwords, client information and other application specific data.

Recommendation:
We recommend implementing a validation mechanism for all the data received from the users.
The best way to protect against SQL Injection is to use prepared statements for every SQL query performed on the database.
Otherwise, the user input can also be sanitized using dedicated methods such as: mysqli_real_escape_string.

More information about SQL injection and the way to protect against this attack can be found here:
https://www.owasp.org/index.php/SQL_Injection
https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.md

File Inclusion

Vulnerable Page	Vulnerable Parameter	Method	Attack Vector	
/dvwa/vulnerabilities/fi/	page	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/fi/?page=%2Fetc%2Fpasswd	🚀
/dvwa/vulnerabilities/fi/	page	GET	http://testing1.pentest-tools.com/dvwa/vulnerabilities/fi/?page=http%3A%2F%2Fwww.google.com%2F	🚀

Details

Risk description:
The risk exists that an attacker can manipulate the affected parameter in order to load or execute any locally or remote stored file.
This could lead to reading arbitrary files, code execution, Cross-Site Scripting, denial of service, sensitive information disclosure.

Recommendation:
The most effective solution to eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API. If this is not possible the application can maintain a white list of files, that may be included by the page, and then use an identifier (for example the index number) to access to the selected file. Any request containing an invalid identifier has to be rejected, in this way there is no attack surface for malicious users to manipulate the path.

References:
https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion
https://www.owasp.org/index.php/Testing_for_Remote_File_Inclusion

OS Command Injection

Vulnerable Page	Vulnerable Parameter	Method	Attack Vector	
/dvwa/vulnerabilities/exec/	ip	POST	http://testing1.pentest-tools.com/dvwa/vulnerabilities/exec/ **POST Data:** ip=ZAP&cat /etc/passwd&	🚀

Details

Risk description:
The risk exists that an attacker uses the application to run OS commands with the privileges of the vulnerable application.
This could lead (but not limited) to Remote Code Execution, Denial of Service, Sensitive Information Disclosure, Sensitive Information Deletion.

Recommendation:
There are multiple ways to mitigate this attack:
- avoid calling OS commands directly (use built-in library functions)
- escape values added to OS commands specific to each OS
- implement parametrization in conjunction with Input Validation (segregate data by command; implement Positive or whitelist input validation; White list Regular Expression)

In order to provide Defense in Depth, we also recommend to allocate the lowest privileges to web applications.

References:
https://www.owasp.org/index.php/Command_Injection
https://www.owasp.org/index.php/OS_Command_Injection_Defense_Cheat_Sheet

Communication is not secure

http://testing1.pentest-tools.com/dvwa/

Details

Risk description:
The communication between the web browser and the server is done using the HTTP protocol, which transmits data unencrypted over the network. Thus, an attacker who manages to intercept the communication at the network level, is able to read and modify the data transmitted (including passwords, secret tokens, credit card information and other sensitive data).

Recommendation:
We recommend you to reconfigure the web server to use HTTPS - which encrypts the communication between the web browser and the server.

Interesting files found

URL	Summary

/dvwa/login.php	Admin login page/section found.
/dvwa/.gitignore	.gitignore file found. It is possible to grasp the directory structure.
/dvwa/config/	Directory indexing found.

Details

Risk description:
These files/folders usually contain sensitive information which may help attackers to mount further attacks against the server. Manual validation is required.

Recommendation:
We recommend you to analyze if the mentioned files/folders contain any sensitive information and restrict their access according to the business purposes of the application.

Server information disclosure

URL	Summary
/dvwa/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000	PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
/dvwa/config/	Configuration information may be available remotely.
/dvwa/docs/	Directory indexing found.
/dvwa/phpinfo.php	PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.

Details

Risk description:
An attacker could use these files to find information about the backend application, server software and their specific versions. This information could be further used to mount targeted attacks against the server.

Recommendation:
We recommend you to remove these scripts if they are not needed for business purposes.

More information about this issue:
http://projects.webappsec.org/w/page/13246936/Information%20Leakage

Server software and technology found

Software / Version	Category
Debian	Operating Systems
Apache 2.4.25	Web Servers

Details

Risk description:
An attacker could use this information to mount specific attacks against the identified software type and version.

Recommendation:
We recommend you to eliminate the information which permit the identification of software platform, technology, server and operating system: HTTP server headers, HTML meta information, etc.

More information about this issue:
https://www.owasp.org/index.php/Fingerprint_Web_Server_(OTG-INFO-002).

Missing HTTP security headers

HTTP Security Header	Header Role	Status
X-Frame-Options	Protects against Clickjacking attacks	Not set

X-XSS-Protection	Mitigates Cross-Site Scripting (XSS) attacks	Not set
X-Content-Type-Options	Prevents possible phishing or XSS attacks	Not set

Details

Risk description:
Because the X-Frame-Options header is not sent by the server, an attacker could embed this website into an iframe of a third party website. By manipulating the display attributes of the iframe, the attacker could trick the user into performing mouse clicks in the application, thus performing activities without user's consent (ex: delete user, subscribe to newsletter, etc). This is called a Clickjacking attack and it is described in detail here:
https://www.owasp.org/index.php/Clickjacking

The X-XSS-Protection HTTP header instructs the browser to stop loading web pages when they detect reflected Cross-Site Scripting (XSS) attacks. Lack of this header exposes application users to XSS attacks in case the web application contains such vulnerability.

The HTTP X-Content-Type-Options header is addressed to Internet Explorer browser and prevents it from reinterpreting the content of a web page (MIME-sniffing) and thus overriding the value of the Content-Type header). Lack of this header could lead to attacks such as Cross-Site Scripting or phishing.

Recommendation:
We recommend you to add the X-Frame-Options HTTP response header to every page that you want to be protected against Clickjacking attacks.
More information about this issue:
https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet

We recommend setting the X-XSS-Protection header to "X-XSS-Protection: 1; mode=block".
More information about this issue:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

We recommend setting the X-Content-Type-Options header to "X-Content-Type-Options: nosniff".
More information about this issue:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

Cookie No HttpOnly Flag

Affected items	Evidence
http://testing1.pentest-tools.com/dvwa/	Set-Cookie: PHPSESSID
http://testing1.pentest-tools.com/dvwa/	Set-Cookie: security
http://testing1.pentest-tools.com/dvwa/vulnerabilities/weak_id/	Set-Cookie: dvwaSession

Details

Risk description:
A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

Recommendation:
Ensure that the HttpOnly flag is set for all cookies.
http://www.owasp.org/index.php/HttpOnly

Robots.txt file not found

No security issue found regarding client access policies

No password input found (clear-text submission test)

No JavaScript vulnerabilities found

No sensitive files found

No server software was identified

No administration consoles were found

Spider results: 28 dynamic URLs of total 50 URLs crawled

ID	METHOD	URL	PARAMS
1	GET	/dvwa/instructions.php	doc=readme
2	POST	/dvwa/setup.php	user_token=9e16aa41c679e13f4cf884b3e2d9ebde&create_db=Create+%2F+Reset+Database
3	POST	/dvwa/login.php	username=ZAP&password=ZAP&user_token=b8a349cc3852ee55ddbafd475d91a33d&Login=Login
4	GET	/dvwa/security.php	phpids=off
5	GET	/dvwa/instructions.php	doc=copying
6	POST	/dvwa/vulnerabilities/exec/	ip=ZAP&Submit=Submit
7	GET	/dvwa/vulnerabilities/xss_d/	default
8	POST	/dvwa/security.php?test=%2522%3E%3Cscript%3Eeval(window.name)%3C/script%3E	security=low&user_token=40c6007dec8ddf0d86245184ac296544&seclev_submit=Submit
9	POST	/dvwa/vulnerabilities/xss_s/	txtName=ZAP&mtxMessage=&btnClear=Clear+Guestbook
10	GET	/dvwa/vulnerabilities/csrf/	Change=Change&password_conf=ZAP&password_new=ZAP
11	GET	/dvwa/vulnerabilities/fi/	page=include.php
12	POST	/dvwa/vulnerabilities/upload/	MAX_FILE_SIZE=100000&uploaded=test_file.txt&Upload=Upload
13	POST	/dvwa/vulnerabilities/xss_s/	txtName=ZAP&mtxMessage=&btnSign=Sign+Guestbook
14	GET	/dvwa/instructions.php	doc=changelog
15	GET	/dvwa/vulnerabilities/brute/	Login=Login&password=ZAP&username=ZAP
16	GET	/dvwa/vulnerabilities/fi/	page=file2.php
17	GET	/dvwa/vulnerabilities/fi/	page=file3.php
18	GET	/dvwa/instructions.php	doc=PHPIDS-license
19	GET	/dvwa/vulnerabilities/sqli/	Submit=Submit&id=ZAP
20	GET	/dvwa/instructions.php	doc=PDF
21	GET	/dvwa/security.php	phpids=on
22	GET	/dvwa/vulnerabilities/xss_r/	name=ZAP
23	GET	/dvwa/vulnerabilities/sqli_blind/	Submit=Submit&id=ZAP
24	GET	/dvwa/security.php	test=%2522%3E%3Cscript%3Eeval(window.name)%3C/script%3E
25	GET	/dvwa/vulnerabilities/fi/	page=file1.php
26	POST	/dvwa/vulnerabilities/captcha/	step=1&password_new=ZAP&password_conf=ZAP&Change=Change
27	POST	/dvwa/security.php	security=low&user_token=080654f0ab97c8c1beab2864c6c8af90&seclev_submit=Submit
28	GET	/dvwa/ids_log.php	clear_log=Clear+Log

Scan coverage information

List of tests performed (19/20)

- ✓ Fingerprinting the server software and technology...
- ✓ Checking for vulnerabilities of server-side software...
- ✓ Analyzing HTTP security headers...
- ✓ Checking for secure communication...
- ✓ Checking robots.txt file...
- ✓ Checking client access policies...
- ✓ Checking for clear-text submission of passwords (quick scan)...
- ✓ Checking for JavaScript vulnerabilities...
- ✓ Searching for sensitive files...
- ✓ Checking for interesting files...
- ✓ Checking for information disclosure...
- ✓ Checking for software identification...
- ✓ Checking for administration consoles...
- ✓ Spidering target...
- ✓ Scanning for XSS vulnerabilities...
- ✓ Scanning for SQL Injection vulnerabilities...
- ✓ Scanning for File Inclusion vulnerabilities...
- ✓ Scanning for OS Command Injection vulnerabilities...
- ✓ Scanning for Cookie No HttpOnly Flag vulnerabilities...

Scan parameters

Website URL:	http://testing1.pentest-tools.com/dvwa/
Scan type:	Full_new
Authentication:	False