wvscanner

Scan for major vulnerabilities in your web application

Presented By

Navid Ishriyaq Amin
Md. Mustaqur Rahman

Supervisor

Md. Masum
Professor
Department of Computer Science and Engineering

Motivation

- Most of the companies does not invest enough time and money to secure their websites.
- Most developers does not dive deep into cybersecurity.
- It would be great for developers to have a easy tool for automated black-box penetration testing.

Our goal

- Our goal was make a simple tool that can scan for more than a dozen of common web security vulnerabilities given an URL
- Generate a report that is easy to understand both developers and administrators.
- Give hints for solving this issues

What is wvscanner?

wvscanner is a terminal based tool. It works as a "black-box" vulnerability scanner. That means it won't study the source code of web applications but will work like a fuzzer.

General features

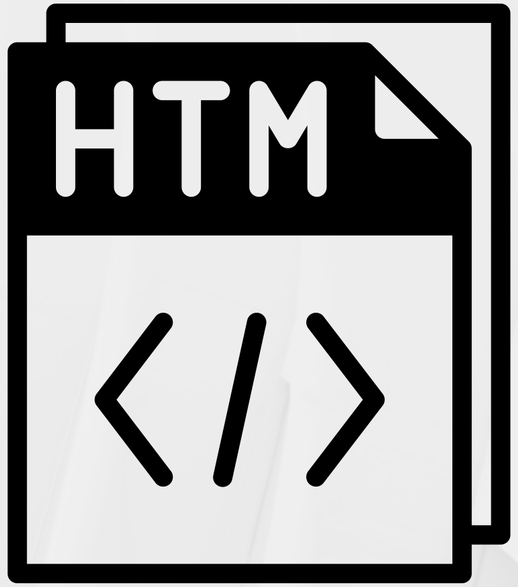
- wvscanner can detect nearly 13 web vulnerabilities.
- Generates vulnerability reports in HTML format.
- Can suspend and resume a scan or an attack (session mechanism using sqlite3 databases)
- Different levels of verbosity.
- Easy to use.

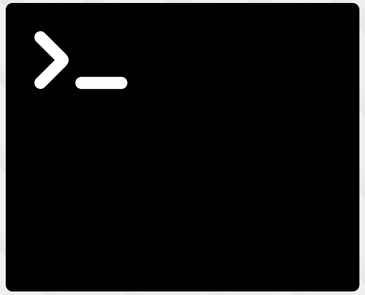
Core functionalities

- Fingerprint web technology
- SQL Injection
- Cross Site Scripting(XSS)
- Cross Site Request Forgery(XSRF)
- Backup file
- Weak credentials
- Content Security Policy Configuration
- HTTP Secure Headers
- HttpOnly Flag cookie
- Secure Flag cookie
- Path Traversal

Though It is a terminal based tool it is very user friendly. Because it can generate whole report with a single command. You can export this report as HTML file.

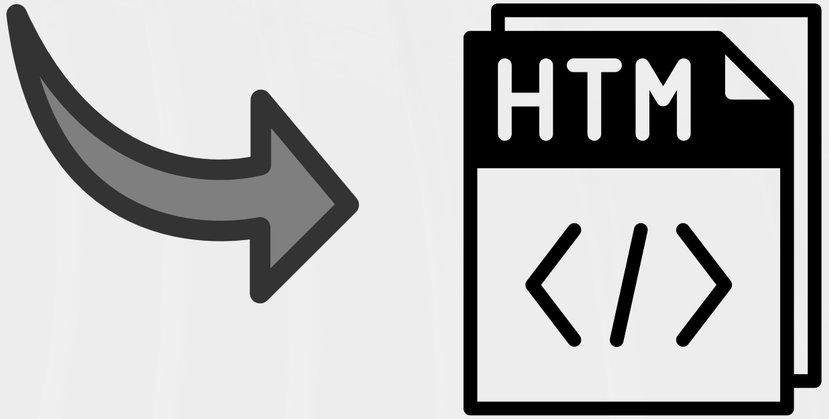
~ $ wvscanner -u url -m all





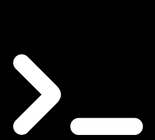
~ $ wvscanner -u url -m sql

~ $ wvscanner -u url -m all



You can run a specific module or all module at once by using -m parameter.

Can suspend and resume a scan.



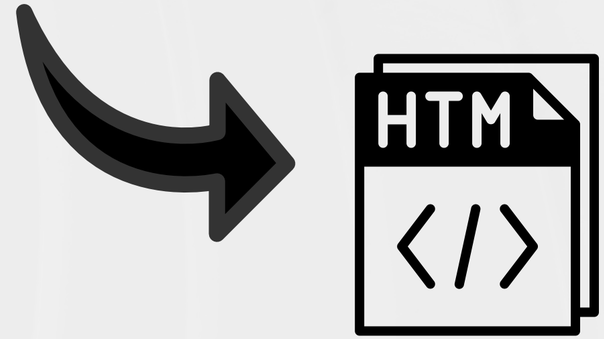
```
Attack process was interrupted. Do you want
to:
 r) stop everything here and generate the
(R)eport
 n) move to the (N)ext attack module (if any)
 q) (Q)uit without generating the report
 c) (C)ontinue the current attack
? n
```

This session mechanism are using sqlite3 databases.

```
Attack process was interrupted. Do you want to:
        r) stop everything here and generate the (R)eport
        n) move to the (N)ext attack module (if any)
        q) (Q)uit without generating the report
        c) (C)ontinue the current attack
? n
```

For demonstration we use **testphp.vulnweb.com/**

```
~ $ wvscanner -u testphp.vulnweb.com/ -m all
```



You can run a specific module or all module at once by using -m parameter.

Scanning Report

Summary

Category	Number of vulnerabilities found
Backup file	2
Weak credentials	1
CRLF Injection	0
Content Security Policy Configuration	1
Cross Site Request Forgery	5
Potentially dangerous file	0
Command execution	0
Path Traversal	4
Fingerprint web application framework	1
Fingerprint web server	1
HTTP Secure Headers	2
HttpOnly Flag cookie	0
Log4Shell	0
Open Redirect	0
Reflected Cross Site Scripting	14
Secure Flag cookie	0
SQL Injection	27
TLS/SSL misconfigurations	0
Server Side Request Forgery	0
Stored Cross Site Scripting	0

Vulnerability found in /secured/newuser.php

Description | HTTP Request

```
POST /secured/newuser.php HTTP/1.1
host: testphp.vulnweb.com
connection: keep-alive
user-agent: Mozilla/5.0 (Windows NT 6.1; rv:45.0) Gecko/20100101 Firefox/45.0
accept-language: en-US
accept-encoding: gzip, deflate, br
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
content-type: application/x-www-form-urlencoded
referer: http://testphp.vulnweb.com/signup.php
content-length: 203
Content-Type: application/x-www-form-urlencoded

uuname=default&upass=Letm3in_&upass2=Letm3in_&urname=default&ucc=default&uemail=wvscanner2021%40mailinator.
```

Solutions

The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

References

- OWASP: Cross Site Scripting (XSS)
- Wikipedia: Cross-site scripting
- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- OWASP: Reflected Cross Site Scripting

- Number of vulnerabilities
- Vulnerability descriptin.
- Some Possible Solution
- References

Server Side Request Forgery	0
Stored Cross Site Scripting	0
Subdomain takeover	0
Blind SQL Injection	0
XML External Entity	0
Internal Server Error	1
Resource consumption	0
Fingerprint web technology	5
HTTP Methods	0

SQL Injection

- Blind SQL Injection
- time based SQL Injection

SQL Injection

Vulnerability found in /userinfo.php

Description | HTTP Request

```
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded

uname=default&pass=%27%09or%09sleep%287%29%231
```

Solutions

To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, user input must be escaped or filtered or parameterized statements must be used.

References

- OWASP: SQL Injection
- Wikipedia: SQL injection
- CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
- OWASP: Blind SQL Injection

Fingerprint web technology

- Fingerprint web application framework
- Fingerprint web server
- Fingerprint web technology

Fingerprint web application framework

Fingerprint web application framework

Description

The version of a web application framework can be identified due to the presence of its specific fingerprints.

INFO Vulnerability found in /

Description | HTTP Request

{"name": "PHP", "versions": ["5.6.40"], "categories": ["Programming languages"], "groups": ["Web developmen

Fingerprint web server

Fingerprint web server

Description

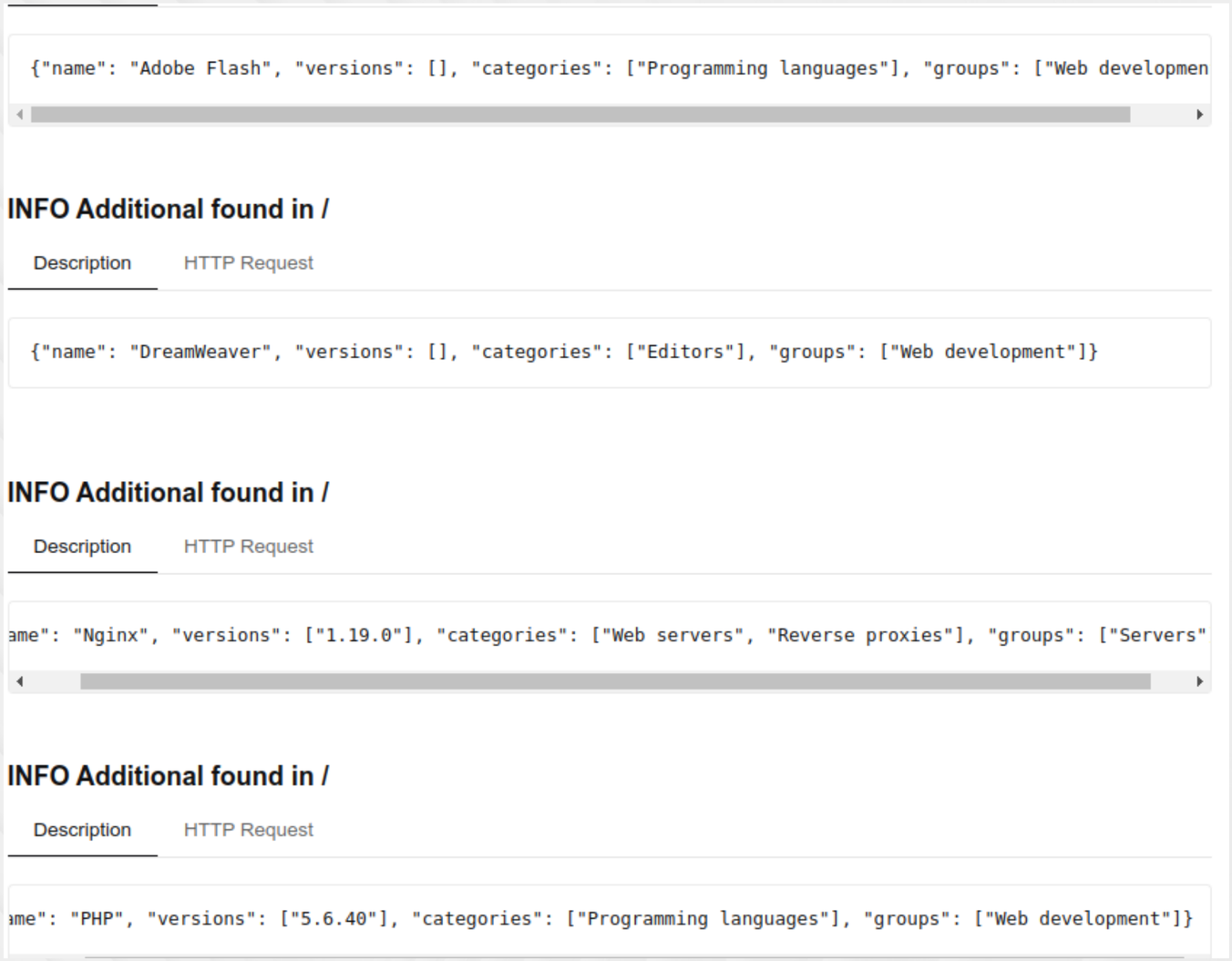
The version of a web server can be identified due to the presence of its specific fingerprints.

INFO Vulnerability found in /

Description | HTTP Request


```
{"name": "Nginx", "versions": ["1.19.0"], "categories": ["Web servers", "Reverse proxies"], "groups": ["Ser
```

Fingerprint web technology



Cross Site Scripting(XSS)

Vulnerability found in /secured/newuser.php

Description | HTTP Request

```
POST /secured/newuser.php HTTP/1.1
host: testphp.vulnweb.com
connection: keep-alive
user-agent: Mozilla/5.0 (Windows NT 6.1; rv:45.0) Gecko/20100101 Firefox/45.0
accept-language: en-US
accept-encoding: gzip, deflate, br
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
content-type: application/x-www-form-urlencoded
referer: http://testphp.vulnweb.com/signup.php
content-length: 203
Content-Type: application/x-www-form-urlencoded

uuname=default&upass=Letm3in_&upass2=Letm3in_&urname=default&ucc=default&
uemail=wvscanner2021%40mailinator.com&uphone=default&signup=signup&uaddress=%3CScRiPt%3Ealert
%28%27w3kgcvw6jz%27%29%3C%2FsCrIpT%3E
```

Solutions

The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -

Cross Site Request Forgery(XSRF)

Cross Site Request Forgery

Description

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

Vulnerability found in /dl_status.php

Description | HTTP Request

```
Lack of anti CSRF token
```

Solutions

Check if your framework has built-in CSRF protection and use it. If framework does not have built-in CSRF protection add CSRF tokens to all state changing requests (requests that cause actions on the site) and validate them on backend.

Backup file

🟡 Vulnerability found in /index.bak

Description | HTTP Request

```
Backup file http://testphp.vulnweb.com/index.bak found for http://testphp.vulnweb.com/index.php
```

🟡 Vulnerability found in /index.zip

Description | HTTP Request

```
Backup file http://testphp.vulnweb.com/index.zip found for http://testphp.vulnweb.com/index.php
```

Weak credentials

Weak credentials

Description

The web application is using either default credentials or weak passwords that can be found in well-known passwords lists.

🟡 Vulnerability found in /userinfo.php

Description | HTTP Request

```
Credentials found for URL http://testphp.vulnweb.com/login.php : test / test
```

Solutions

Do not ship or deploy with any default credentials, particularly for admin users. Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.

Content Security Policy Configuration

Content Security Policy Configuration

Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.

🟡 Vulnerability found in /

Description | HTTP Request

```
CSP is not set
```

Solutions

Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

Path Traversal

Vulnerability found in /showimage.php

Description | HTTP Request

```
Possible fopen() vulnerability via injection in the parameter file
```

Vulnerability found in /showimage.php

Description | HTTP Request

```
GET /showimage.php?file=%2Fetc%2Fpasswd&size=160 HTTP/1.1
host: testphp.vulnweb.com
connection: keep-alive
user-agent: Mozilla/5.0 (Windows NT 6.1; rv:45.0) Gecko/20100101 Firefox/45.0
accept-language: en-US
accept-encoding: gzip, deflate, br
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
```

What we could not implement

- Log4shell
- Shellshock
- Full Graphical User Interface.

Thank you.