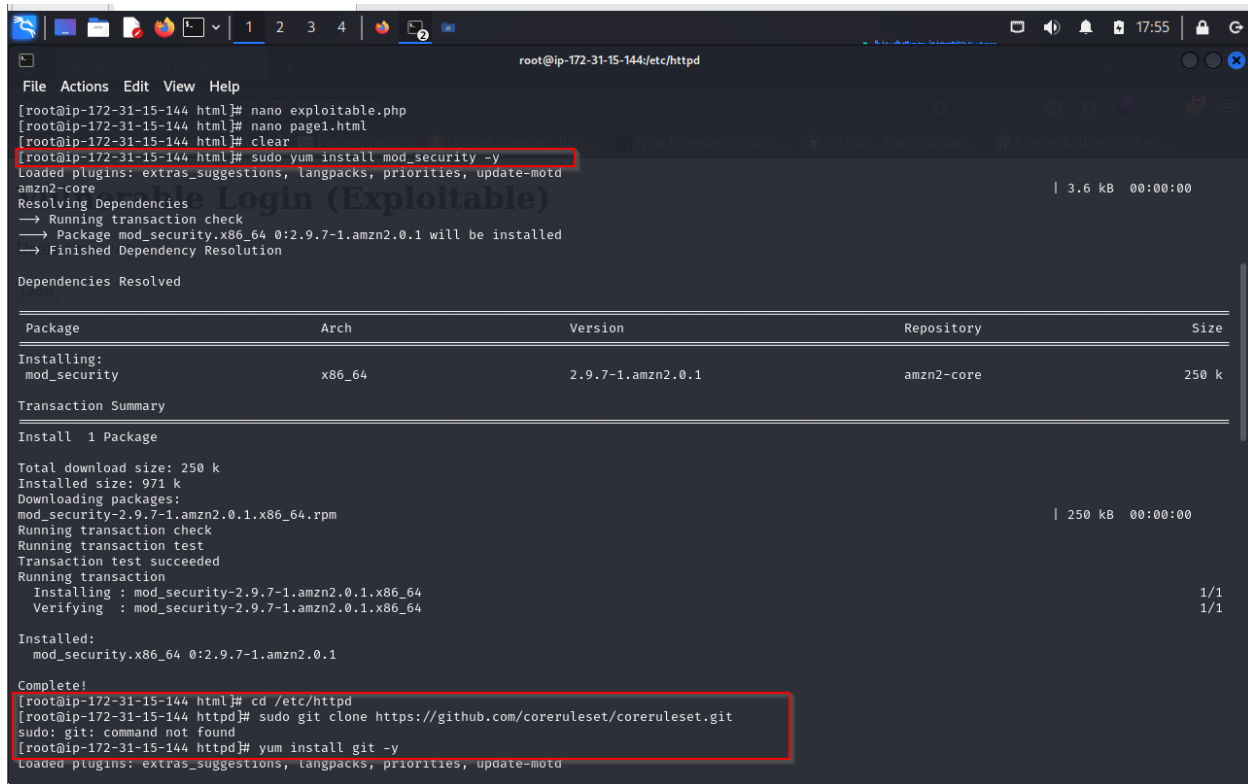


## Part 2: Mitigate with ModSecurity WAF

### 2.1 Install & Enable ModSecurity

1. Install ModSecurity (for Apache on Amazon Linux 2):

```
sudo yum install mod_security -y
```

A terminal window screenshot showing the installation of mod\_security. The user is in a shell as root on ip-172-31-15-144. They run 'sudo yum install mod\_security -y'. The terminal shows the package being installed from the 'amzn2-core' repository. A table lists the package details: mod\_security, x86\_64, version 2.9.7-1.amzn2.0.1, size 250 k. The transaction summary shows the package is installed successfully. The user then runs 'cd /etc/httpd' and 'sudo git clone https://github.com/coreruleset/coreruleset.git'. The terminal shows the git command is not found, and then 'yum install git -y' is run to install git.

```
root@ip-172-31-15-144/etc/httpd
File Actions Edit View Help
[root@ip-172-31-15-144 html]# nano exploitable.php
[root@ip-172-31-15-144 html]# nano page1.html
[root@ip-172-31-15-144 html]# clear
[root@ip-172-31-15-144 html]# sudo yum install mod_security -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.6 kB 00:00:00
Resolving Dependencies
--> Running transaction check
--> Package mod_security.x86_64 0:2.9.7-1.amzn2.0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

Package Arch Version Repository Size
--
Installing:
mod_security x86_64 2.9.7-1.amzn2.0.1 amzn2-core 250 k

Transaction Summary
--
Install 1 Package

Total download size: 250 k
Installed size: 971 k
Downloading packages:
mod_security-2.9.7-1.amzn2.0.1.x86_64.rpm | 250 kB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : mod_security-2.9.7-1.amzn2.0.1.x86_64 1/1
Verifying : mod_security-2.9.7-1.amzn2.0.1.x86_64 1/1

Installed:
mod_security.x86_64 0:2.9.7-1.amzn2.0.1

Complete!
[root@ip-172-31-15-144 html]# cd /etc/httpd
[root@ip-172-31-15-144 httpd]# sudo git clone https://github.com/coreruleset/coreruleset.git
sudo: git: command not found
[root@ip-172-31-15-144 httpd]# yum install git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
```

2. Configure /etc/httpd/conf.d/mod\_security.conf (or similar):

```
SecRuleEngine On
```

3. OWASP Core Rule Set:

```
cd /etc/httpd
```

```
sudo git clone https://github.com/coreruleset/coreruleset.git
```

```
# In your mod_security.conf:
```

```
# IncludeOptional /etc/httpd/coreruleset/crs-setup.conf
```

```
# IncludeOptional /etc/httpd/coreruleset/rules/*.conf
```

```
root@ip-172-31-15-144:/etc/httpd
File Actions Edit View Help
GNU nano 2.9.8 /etc/httpd/conf.d/mod_security.conf Modified

PE %{REQBODY_PROCESSOR_ERROR}, \
BQ %{MULTIPART_BOUNDARY_QUOTED}, \
BW %{MULTIPART_BOUNDARY_WHITESPACE}, \
DB %{MULTIPART_DATA_BEFORE}, \
DA %{MULTIPART_DATA_AFTER}, \
HF %{MULTIPART_HEADER_FOLDING}, \
LF %{MULTIPART_LF_LINE}, \
SM %{MULTIPART_MISSING_SEMICOLON}, \
IQ %{MULTIPART_INVALID_QUOTING}, \
IP %{MULTIPART_INVALID_PART}, \
IH %{MULTIPART_INVALID_HEADER_FOLDING}, \
FL %{MULTIPART_FILE_LIMIT_EXCEEDED}"

SecRule MULTIPART_UNMATCHED_BOUNDARY "!@eq 0" \
  "id:'200003',phase:2,t:none,log,deny,status:44,msg:'Multipart parser detected a possible unmatched boundary.'"

SecPcreMatchLimit 1000
SecPcreMatchLimitRecursion 1000

SecRule TX:/"MSC_/"!@streq 0" \
  "id:'200004',phase:2,t:none,deny,msg:'ModSecurity internal error flagged: %{MATCHED_VAR_NAME}'"

SecResponseBodyAccess Off
SecDebugLog /var/log/httpd/modsec_debug.log
SecDebugLogLevel 0
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?:?!04))"
SecAuditLogParts ABIJDEFHZ
SecAuditLogType Serial
SecAuditLog /var/log/httpd/modsec_audit.log
SecArgumentSeparator &
SecCookieFormat 0
SecTmpDir /var/lib/mod_security
SecDataDir /var/lib/mod_security
IncludeOptional /etc/httpd/coreruleset/crs-setup.conf
IncludeOptional /etc/httpd/coreruleset/rules/*.conf
</IfModule>

G Get Help W Write Out W Where Is C Cut Text J Justify C Cur Pos M-U Undo M-A Mark Text M-] To Bracket
X Exit R Read File R Replace U Uncut Text T To Spell G Go To Line M-E Redo M-C Copy Text M-^ WhereIs Next
```

#### 4.Restart Apache:

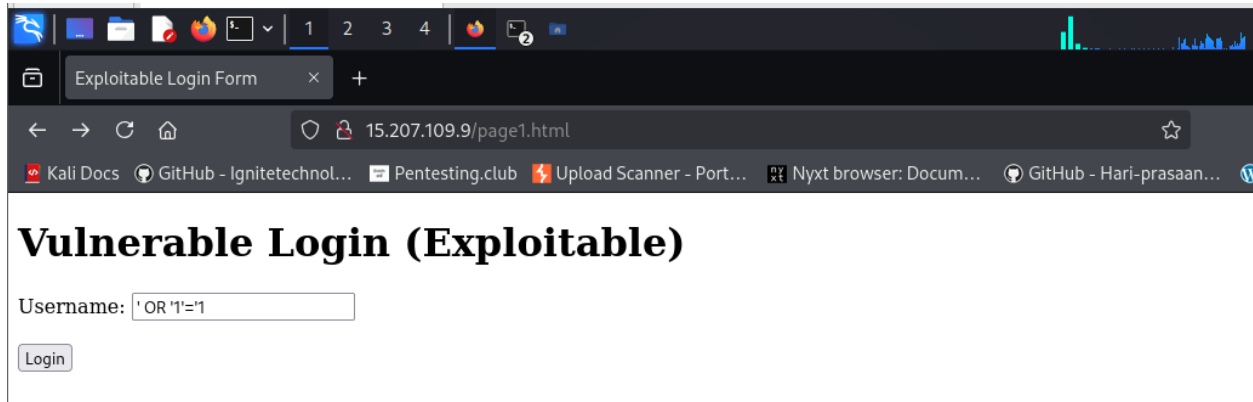
sudo systemctl restart httpd

```
Complete!
[root@ip-172-31-15-144 httpd]# sudo git clone https://github.com/coreruleset/coreruleset.git
Cloning into 'coreruleset'...
remote: Enumerating objects: 34447, done.
remote: Counting objects: 100% (206/206), done.
remote: Compressing objects: 100% (79/79), done.
remote: Total 34447 (delta 179), reused 127 (delta 127), pack-reused 34241 (from 5)
Receiving objects: 100% (34447/34447), 9.87 MiB | 22.55 MiB/s, done.
Resolving deltas: 100% (27181/27181), done.
[root@ip-172-31-15-144 httpd]# nano /etc/httpd/conf.d/mod_security.conf
[root@ip-172-31-15-144 httpd]# ls
conf  conf.d  conf.modules.d  coreruleset  logs  modsecurity.d  modules  run  state
[root@ip-172-31-15-144 httpd]# cd coreruleset
[root@ip-172-31-15-144 coreruleset]# ls
CHANGES.md  CONTRIBUTORS.md  docs  KNOWN_BUGS.md  plugins  regex-assembly  rules  SPONSORS.md  util
CONTRIBUTING.md  crs-setup.conf.example  INSTALL.md  LICENSE  README.md  renovate.json  SECURITY.md  tests
[root@ip-172-31-15-144 coreruleset]# mv crs-setup.conf.example crs-setup.conf
[root@ip-172-31-15-144 coreruleset]# ls
CHANGES.md  CONTRIBUTORS.md  docs  KNOWN_BUGS.md  plugins  regex-assembly  rules  SPONSORS.md  util
CONTRIBUTING.md  crs-setup.conf  INSTALL.md  LICENSE  README.md  renovate.json  SECURITY.md  tests
[root@ip-172-31-15-144 coreruleset]#
```

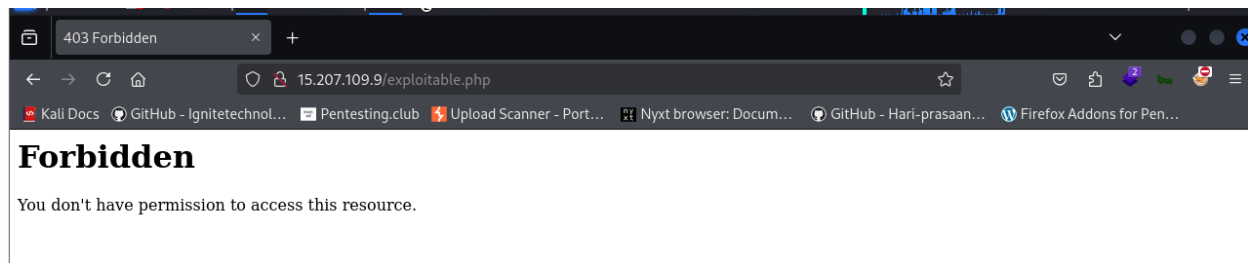
```
root@ip-172-31-15-144:/etc/httpd/coreruleset
File Actions Edit View Help
[root@ip-172-31-15-144 coreruleset]# sudo systemctl restart httpd
[root@ip-172-31-15-144 coreruleset]#
```

## 2.2 Verify Blocking of SQL Injection

1. Return to <http://15.207.109.9/page1.html>
2. Enter the same injection payload: ' OR '1'='1.



3. If ModSecurity is configured correctly, you should see a 403 Forbidden or an error page indicating the request was blocked.



## 2.3 Create the Non-Exploitable Form (page2.html + secure.php)

We'll also demonstrate secure code using parameterized queries.

1. page2.html (Non-Exploitable Form)

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <title>Secure Login Form</title>
```

```
</head>

<body>

  <h1>Secure Login (Non-Exploitable)</h1>

  <form method="POST" action="secure.php">

    <label for="username">Username:</label>

    <input type="text" id="username" name="username">

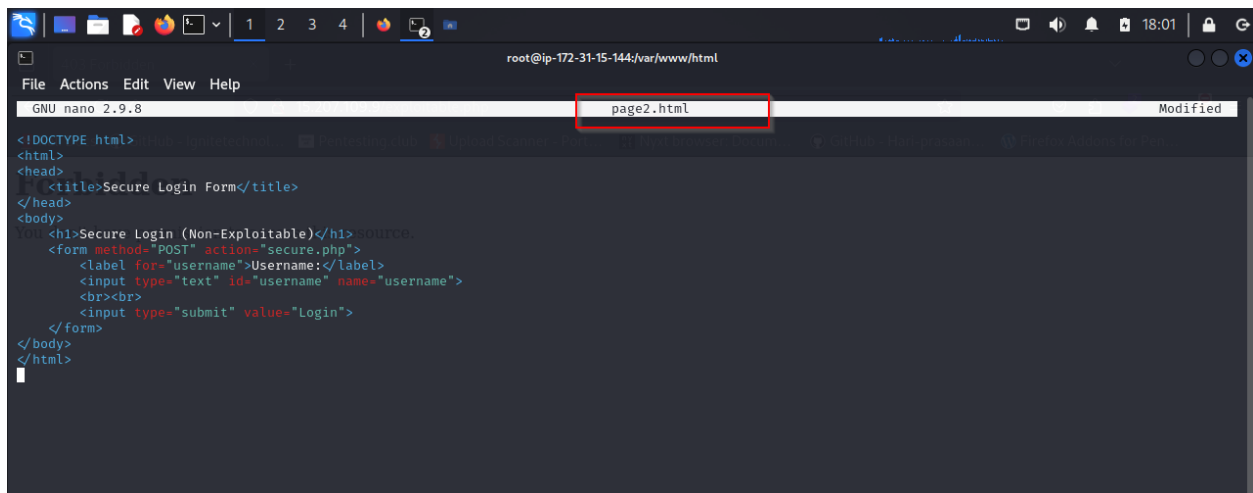
    <br><br>

    <input type="submit" value="Login">

  </form>

</body>

</html>
```



The screenshot shows a terminal window with the nano text editor open. The editor is editing a file named 'page2.html'. The code visible in the editor is the same HTML code as shown in the previous block. The terminal window title is 'root@ip-172-31-15-144:/var/www/html'. The nano editor's status bar at the top shows 'GNU nano 2.9.8' and 'page2.html Modified'.

## 2. secure.php (Using parameterized queries)

```
<?php
// secure.php

$host = "localhost";
$dbname = "injection_demo";
$user = "root";
$pass = "YOUR_DB_ROOT_PASSWORD"; // Replace with actual password

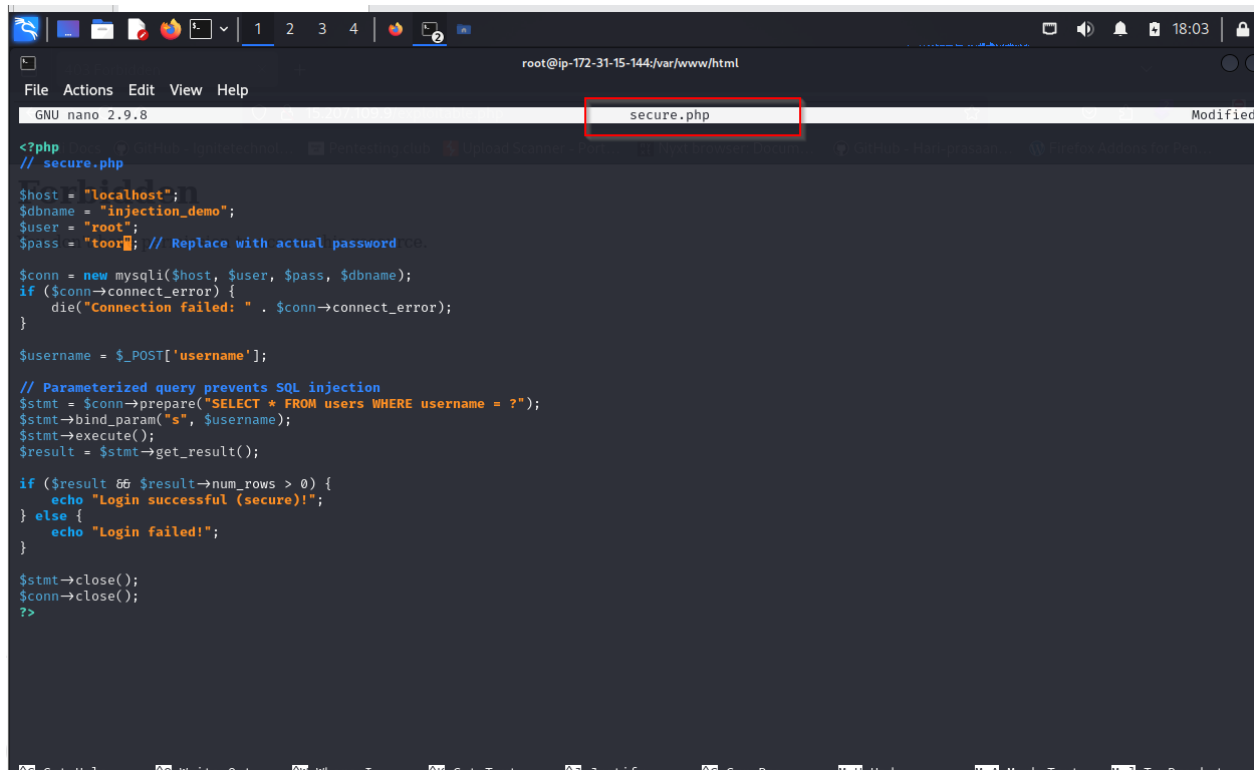
$conn = new mysqli($host, $user, $pass, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$username = $_POST['username'];

// Parameterized query prevents SQL injection
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ?");
$stmt->bind_param("s", $username);
$stmt->execute();
$result = $stmt->get_result();

if ($result && $result->num_rows > 0) {
    echo "Login successful (secure)!" ;
} else {
    echo "Login failed!";
}

$stmt->close();
$conn->close();
?>
```



```
GNU nano 2.9.8 secure.php
<?php
// secure.php

$host = "localhost";
$dbname = "injection_demo";
$user = "root";
$pass = "toor"; // Replace with actual password DB.

$conn = new mysqli($host, $user, $pass, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$username = $_POST['username'];

// Parameterized query prevents SQL injection
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ?");
$stmt->bind_param("s", $username);
$stmt->execute();
$result = $stmt->get_result();

if ($result && $result->num_rows > 0) {
    echo "Login successful (secure)!";
} else {
    echo "Login failed!";
}

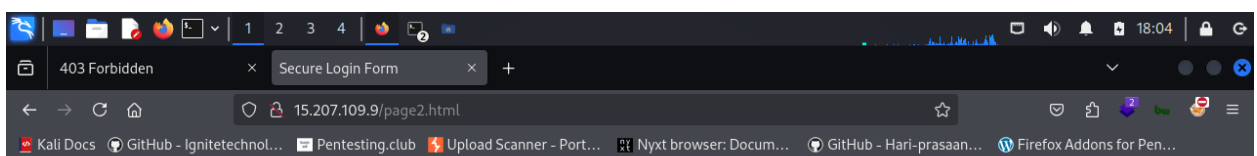
$stmt->close();
$conn->close();
?>
```

Location: Place page2.html and secure.php in /var/www/html/.

URL: <http://15.207.109.9/page2.html>

Injection Test: ' OR '1'='1

Should fail to log in because of parameterized queries (and also possibly blocked by ModSecurity).



## Secure Login (Non-Exploitable)

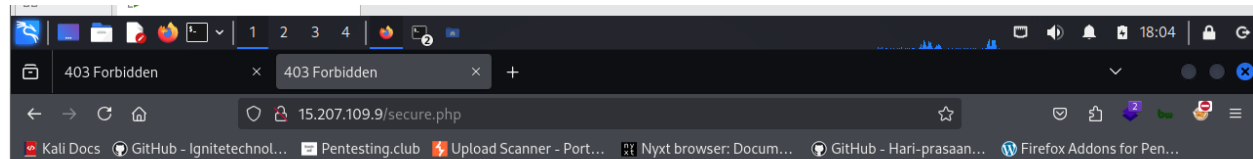
Username:



## Secure Login (Non-Exploitable)

Username:

Login



## Forbidden

You don't have permission to access this resource.

**NOTE: For making the page1.html vulnerable again Temporarily Disable or Bypass ModSecurity (if active)**

If ModSecurity is currently enabled on your server, it might block SQL injection attempts. To test the vulnerable page, you can disable ModSecurity temporarily:

- **Disable ModSecurity Globally (for testing only)** Edit your ModSecurity configuration in `/etc/httpd/conf.d/mod_security.conf` and set:

`SecRuleEngine Off`

```
root@ip-172-31-15-144:/home/ec2-user
File Actions Edit View Help
GNU nano 2.9.8 /etc/httpd/conf.d/mod_security.conf

# IfModule mod_security2.c>
# ModSecurity Core Rules Set configuration
IncludeOptional modsecurity.d/*.conf
IncludeOptional modsecurity.d/activated_rules/*.conf

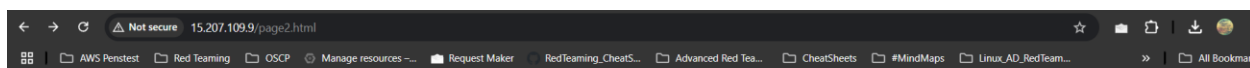
# Default recommended configuration
SecRuleEngine Off
SecRequestBodyAccess On
SecRule REQUEST_HEADERS:Content-Type "text/xml" \
  "id: '200000', phase:1, t:none, t:lowercase, pass, nolog, ctl:requestBodyProcessor=XML"
SecRequestBodyLimit 13107200
SecRequestBodyNoFilesLimit 131072
SecRequestBodyInMemoryLimit 131072
SecRequestBodyLimitAction Reject
SecRule REQBODY_ERROR "!@eq 0" \
  "id: '200001', phase:2, t:none, log, deny, status:400, msg: 'Failed to parse request body.', logdata: '%{reqbody_error_msg}', severity:2"
SecRule MULTIPART_STRICT_ERROR "!@eq 0" \
  "id: '200002', phase:2, t:none, log, deny, status:44, msg: 'Multipart request body \
  failed strict validation: \
  PE %{REQBODY_PROCESSOR_ERROR}, \
  BQ %{MULTIPART_BOUNDARY_QUOTED}, \
  BW %{MULTIPART_BOUNDARY_WHITESPACE}, \
  DB %{MULTIPART_DATA_BEFORE}, \
  DA %{MULTIPART_DATA_AFTER}, \
  HF %{MULTIPART_HEADER_FOLDING}, \
  LF %{MULTIPART_LF_LINE}, \
  SM %{MULTIPART_MISSING_SEMICOLON}, \
  IQ %{MULTIPART_INVALID_QUOTING}, \
  IP %{MULTIPART_INVALID_PART}, \
  IH %{MULTIPART_INVALID_HEADER_FOLDING}, \
  FL %{MULTIPART_FILE_LIMIT_EXCEEDED}'"
SecRule MULTIPART_UNMATCHED_BOUNDARY "!@eq 0" \
  "id: '200003', phase:2, t:none, log, deny, status:44, msg: 'Multipart parser detected a possible unmatched boundary.'"

SecPcreMatchLimit 1000
SecPcreMatchLimitRecursion 1000
```

Then restart Apache:

```
sudo systemctl restart httpd
```

Even the SecRuleEngine Off but due to parameterized queries the sql injection wont work in page2.html.



## Secure Login (Non-Exploitable)

Username:



