Day 4 : Cybersecurity

PortSwigger:

Access Control Labs:

1) User role controlled by request parameter

Objective:

Exploit a vulnerability where user roles are controlled by a request parameter to escalate privileges and gain access to restricted areas.

- **Vulnerability Name**: User Role Controlled by Request Parameter
- **IP Address (Vulnerable Machine)**: *192.168.47.129 of the vulnerable machine*
- **Severity of Vulnerability**: High (since an attacker could escalate privileges by tampering with role information in requests)
- **Impacts**: Unauthorized access to resources, privilege escalation

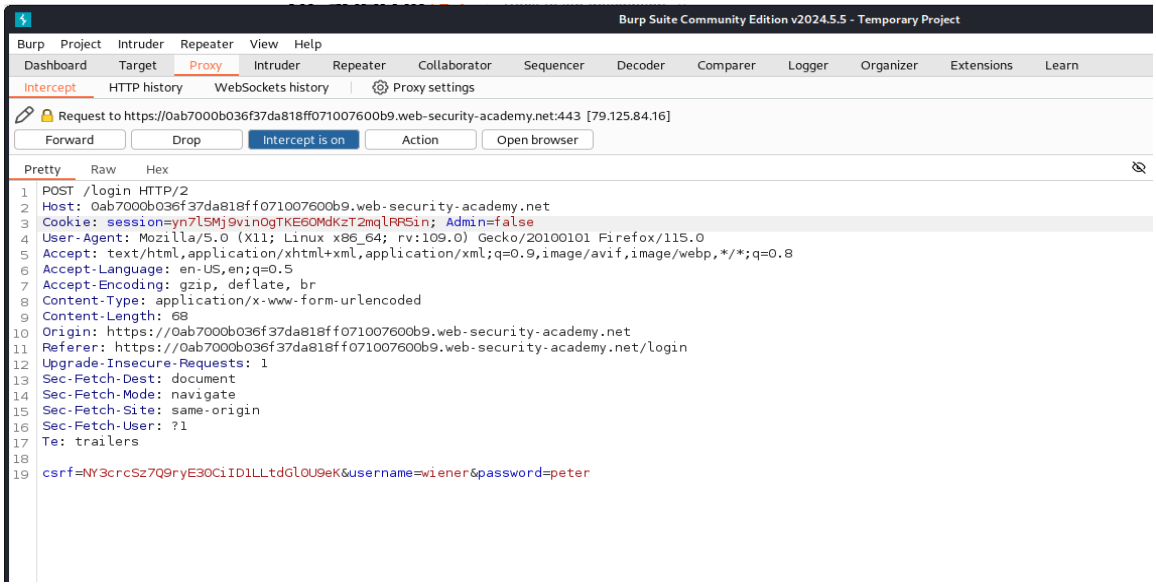
Steps:

1. **Login and Intercept:**
   Log in as a normal user and intercept traffic using Burp Suite.

2. **Identify Role Parameter:**
   Look for a parameter in the requests controlling the role (e.g., role=user).

3. **Modify the Request:**
   Change the role from user to admin (e.g., role=admin) and forward the request.

4. **Verify Privilege Escalation:**
   Confirm access to admin functionalities after role modification.



Web Security Academy

User role controlled by request parameter

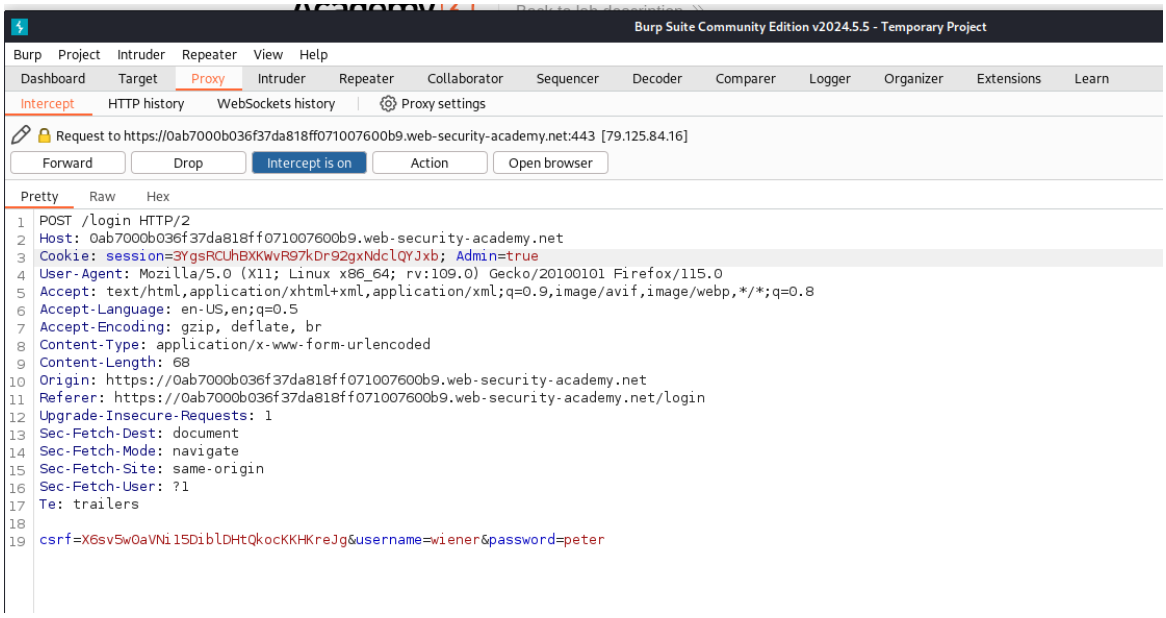
Back to lab description

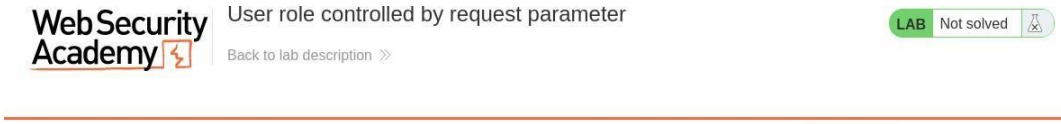
My Account

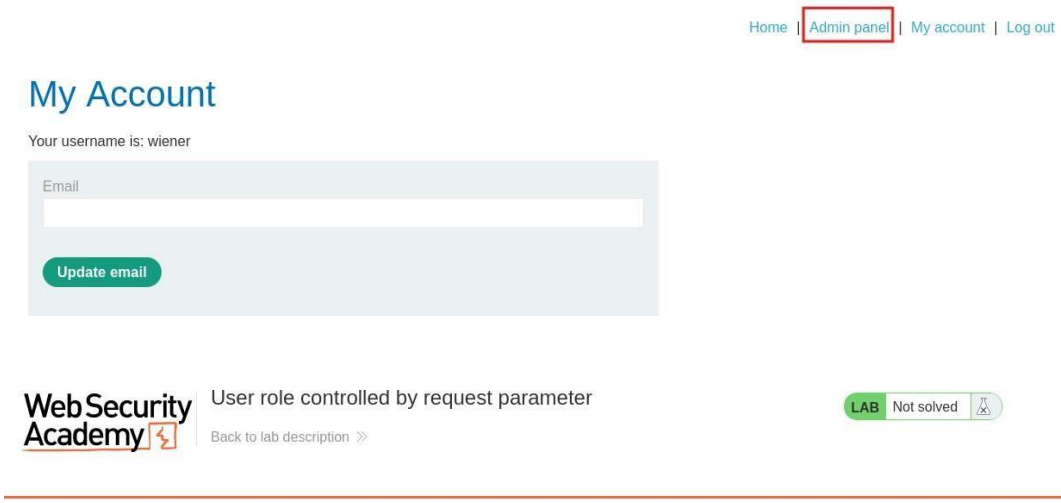
Your username is: wiener

Email

Update email


Burp Suite Community Edition v2024.5.5 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Request to https://0ab7000b036f37da818ff07100760b9.web-security-academy.net:443 [79.125.84.16]
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 POST /login HTTP/2
2 Host: 0ab7000b036f37da818ff07100760b9.web-security-academy.net
3 Cookie: session=3YgsRCUhBXKWvR97kDr92gxNdclQYJxb; Admin=true
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 68
10 Origin: https://0ab7000b036f37da818ff07100760b9.web-security-academy.net
11 Referer: https://0ab7000b036f37da818ff07100760b9.web-security-academy.net/login
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18
19 csrf=X6sv5w0aVNi15DiblDHtQkocKKHKreJg&username=wiener&password=peter


Web Security Academy
User role controlled by request parameter
Back to lab description
LAB Not solved


Home | Admin panel | My account | Log out
My Account
Your username is: wiener
Email
Update email

Web Security Academy
User role controlled by request parameter
Back to lab description
LAB Not solved


Home | Admin panel | My account
Users
wiener - Delete
carlos - Delete

Figure: *Output Result*

2) User ID controlled by request parameter, with unpredictable user IDs

Objective:

Exploit a vulnerability where user IDs are controlled by request parameters, despite the IDs being unpredictable, to access another user's account.

- **Vulnerability Name:** User ID Controlled by Request Parameter (Unpredictable)
- **IP Address (Vulnerable Machine):** *192.168.47.129*
- **Severity of Vulnerability:** Moderate (depends on the strength of the user ID generation)
- **Impacts:** Exposure of sensitive information, unauthorized access to user accounts

Hover over Carlos' post:

https://0a1600010313b6ff8011947d00e300de.web-security-academy.net/blogs?userId=c03dd3c8-ffc7-4296-a251-6ff7a9c541f3

Change user id of wiener to that of carlos while logging in:

User id of carlos =c03dd3c8-ffc7-4296-a251-6ff7a9c541f3

My Account

Your username is: carlos

Your API Key is: 569ShMc1vwTKad2nghlv8zV9ygk6JgsF

Email

Update email

Congratulations, you solved the lab!

Share your skills!

Continue learning

Home | My account | Log out

My Account

Your username is: wiener

Your API Key is: IFyGDThKutBxuCVJ9EdPU9fe4C6SC8RM

Email

Update email

3) Insecure direct object references

Objective:

To identify and exploit Insecure Direct Object References (IDOR) vulnerabilities, where attackers can manipulate object references (like user IDs) to access unauthorized data.

- **Vulnerability Name**: Insecure Direct Object References (IDOR)
- **IP Address (Vulnerable Machine)**: *192.168.47.129*
- **Severity of Vulnerability**: High
- **Impacts**: Unauthorized access to data or resources

Steps:

1. Understand IDOR:

- Occurs when sensitive objects (like user profiles or files) are accessed without proper authorization.

2. Identify Vulnerable Areas:

- Look for object references (e.g., IDs or filenames) in URLs or API endpoints.

3. Test for Vulnerabilities:

- Modify object references (e.g., change `123` to `124` in a URL) to see if you gain access to unauthorized data.

4. Remediation:

- Recommend implementing strict access controls and using indirect references to prevent exploitation.

Web Security Academy | Insecure direct object references

Back to lab description

LAB Not solved

Home | My account | Live chat

Live chat

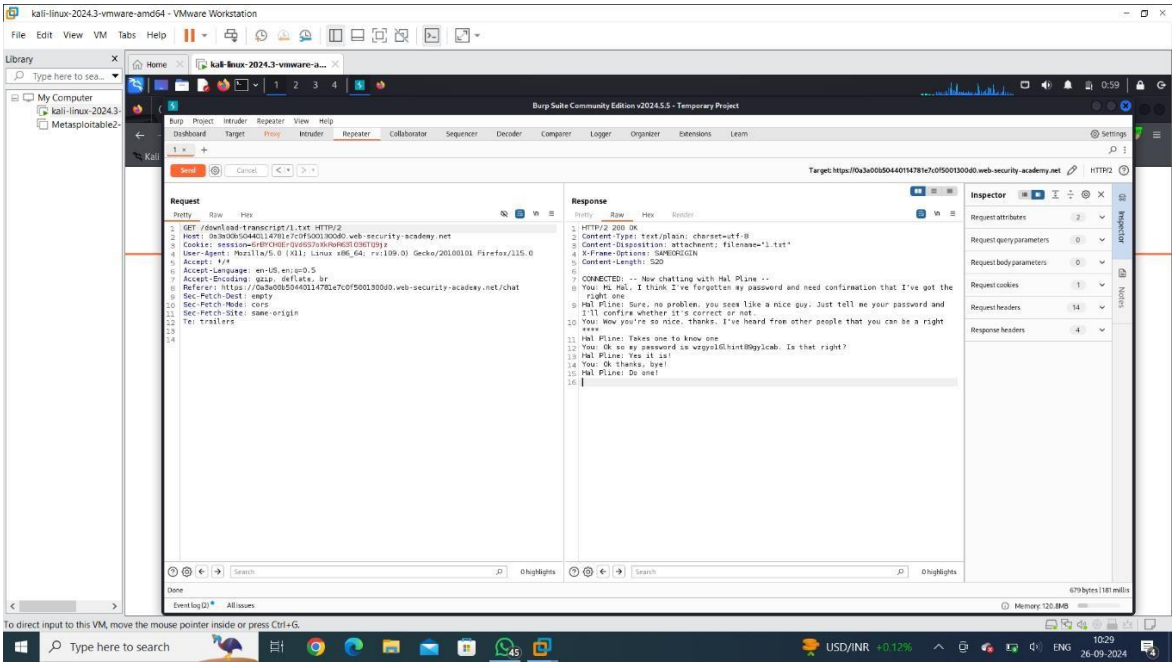
System: No chat history on record

CONNECTED: -- Now chatting with Hal Pline --

Your message:

Hi , I am under the water

Send View transcript



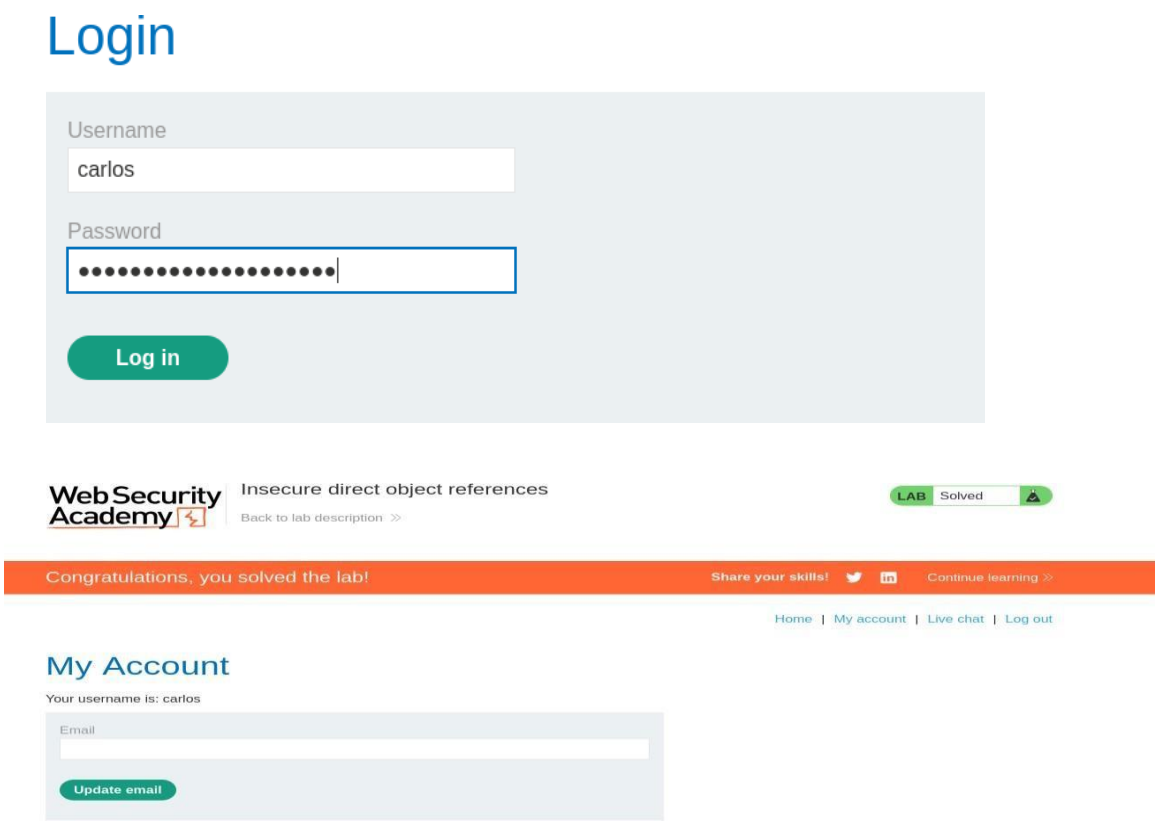


Figure: *Output Result*

4) Reflected XSS into HTML context with nothing encoded

Objective:

To understand and exploit Reflected Cross-Site Scripting (XSS) vulnerabilities where user input is reflected into an HTML context without any encoding, allowing malicious scripts to be executed.

- **Vulnerability Name**: Reflected XSS in HTML Context
- **IP Address (Vulnerable Machine)**: *192.168.47.129*
- **Severity of Vulnerability**: Critical
- **Impacts**: Execution of arbitrary scripts in a victim's browser, theft of sensitive information (e.g., session cookies)

Steps:

1. Understand Reflected XSS:

- Occurs when user-supplied input is immediately reflected in a web page and executed in the browser without being properly sanitized or encoded.

2. Identify Vulnerable Input Fields:

- Look for areas where user input is reflected in the HTML response, such as search fields or query parameters.

3. Test for XSS Vulnerabilities:

- Inject a simple XSS payload (e.g., `<script>alert(1)</script>`) into input fields or URLs.

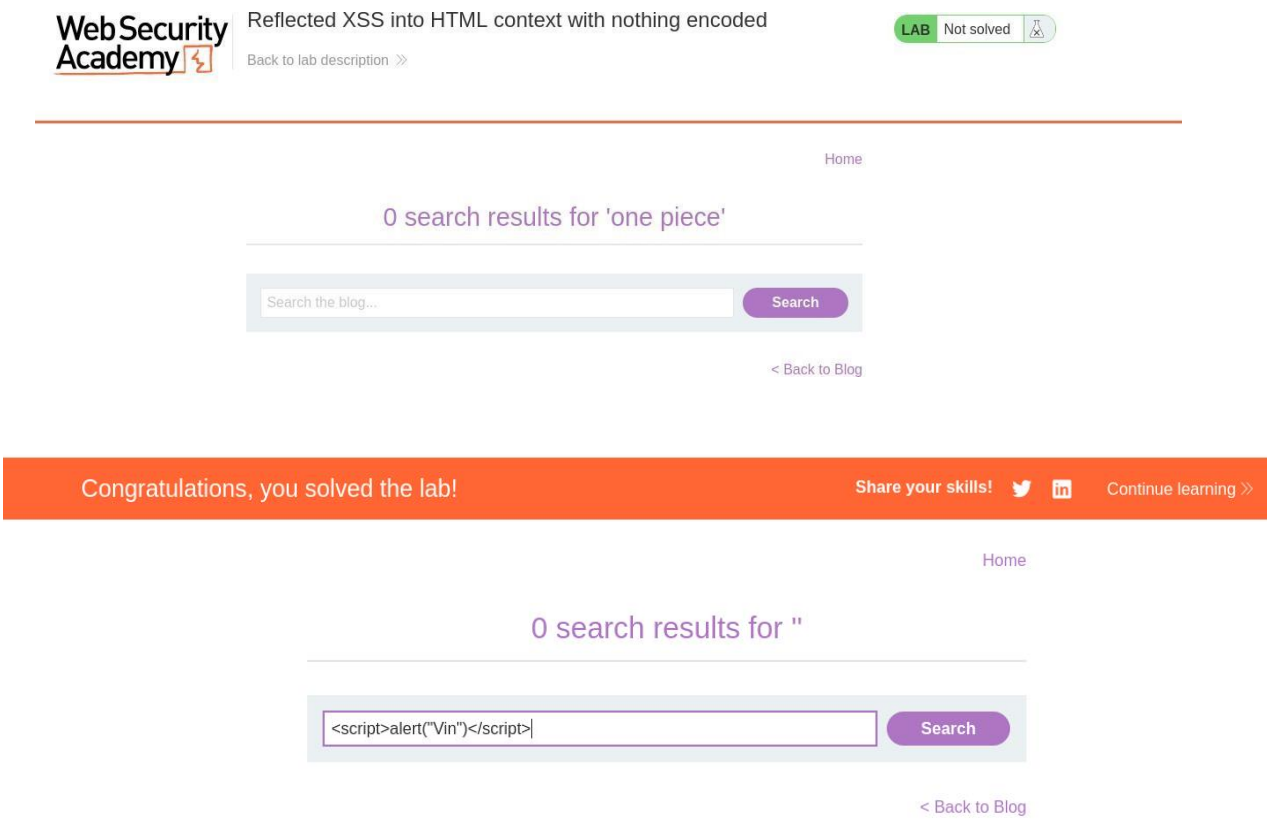
- Example:

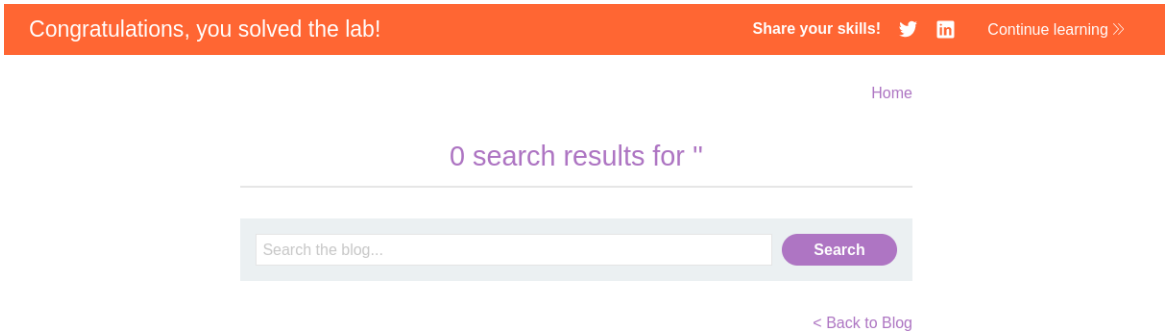
https://example.com/search?query=<script>alert(1)</script>

- Check if the script executes in the browser, confirming the vulnerability.

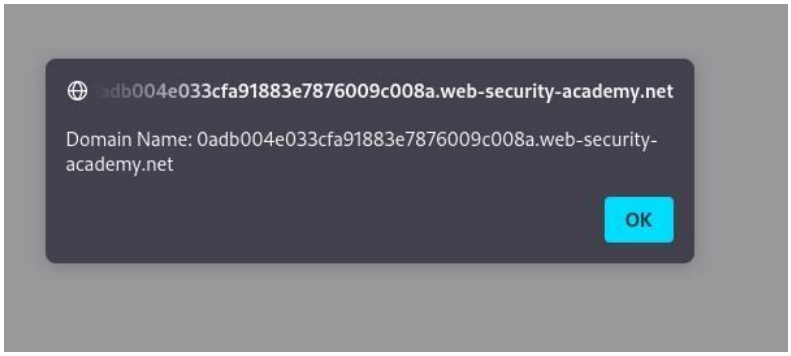
4. Remediation:

- Recommend proper input sanitization and output encoding to prevent script execution.





To get Hostname through script:



5) Stored XSS into HTML context with nothing encoded

Objective:

To understand and exploit Stored Cross-Site Scripting (XSS) vulnerabilities, where malicious input is stored on the server and later displayed in an HTML context without encoding, allowing script execution when accessed by other users.

- **Vulnerability Name:** Stored XSS in HTML Context
- **IP Address (Vulnerable Machine):** *192.168.47.129*
- **Severity of Vulnerability:** Critical
- **Impacts:** Persistent execution of arbitrary scripts in a victim's browser, potentially affecting multiple users

Steps:

1. Understand Stored XSS:

- Occurs when malicious scripts are stored in the database (e.g., in comments or profiles) and executed when accessed by other users.

2. Identify Input Points:

- Look for areas where user input is stored (e.g., comments, forums, or profiles) and reflected back in the HTML context.

3. Test for Vulnerabilities:

- Inject a malicious XSS payload (e.g., `<script>alert('til')</script>`) into the input field.

- Wait for the script to execute when the page is revisited by yourself or others, confirming the stored XSS.

4. Remediation:

- Suggest output encoding for any user-supplied data and input validation to prevent script injections.

Feedback

Our Frequently Asked Questions area will help you with many of your inquiries.
If you can't find your question, return to this page and use the e-mail form below.

IMPORTANT! This feedback facility is not secure. Please do not send any account information in a message sent from here.

To: **Online Banking**

Your Name: Admin User

Your Email Address:

Subject:

Question/Comment: <script>alert("Vineet")</script>

Submit Clear Form

Inc.

Congratulations, you solved the lab! Share your skills! Continue learning »

Home

Thank you for your comment!

Your comment has been submitted.

< Back to blog

6) OM XSS in document.write sink using source location.search

Objective:

To identify and exploit DOM-based Cross-Site Scripting (XSS) vulnerabilities that occur when user input is directly manipulated by JavaScript (in this case, via `document.write()`) without encoding or validation, allowing malicious code execution.

- **Vulnerability Name**: DOM-based XSS via document.write
- **IP Address (Vulnerable Machine)**: *192.168.47.129*
- **Severity of Vulnerability**: High
- **Impacts**: Execution of arbitrary scripts in a victim's browser through manipulation of the DOM

Steps:

1. Understand DOM XSS:

- Occurs when user-controlled data (e.g., `location.search`) is directly inserted into the DOM without proper sanitization.

- In this case, the data is written to the page via `document.write()`.

2. Identify Vulnerable JavaScript Code:

- Look for JavaScript that reads data from `location.search` (e.g., URL parameters) and writes it to the DOM using `document.write()`.

- Example:

document.write(location.search);

3. Test for Vulnerabilities:

- Inject an XSS payload into the URL's query string:

https://example.com/?param=<script>alert(1)</script>

- If the script executes, the vulnerability is confirmed.

4. Remediation:

- Recommend avoiding the use of `document.write()` and sanitizing/encoding all user-controlled input before inserting it into the DOM.

0 search results for '"><script>alert("Hello")</script

"><script>alert("Hello")</script>

Search

">

< Back t

Web Security Academy

DOM XSS in `document.write` sink using source `location.search`

LAB Solved

Back to lab description »

Congratulations, you solved the lab!

Share your skills!

Continue learning »

Home

0 search results for '"><script>alert("Hello");</script>'

Search the blog...

Search

">

< Back to Blog

Command Execution Vulnerability (DVWA)

-Command Execution vulnerability being exploited in DVWA. The user input a target IP address (192.168.47.129) in the "Ping for FREE" field.

- The results display the output of the `ping` command, confirming that the command is executed on the server and sent ICMP packets to the given IP address, successfully returning responses from the target.

- Additionally, an extra input " cat tilak.txt " was appended.

Critical severity: *Command line (code) execution*

Steps to replicate:

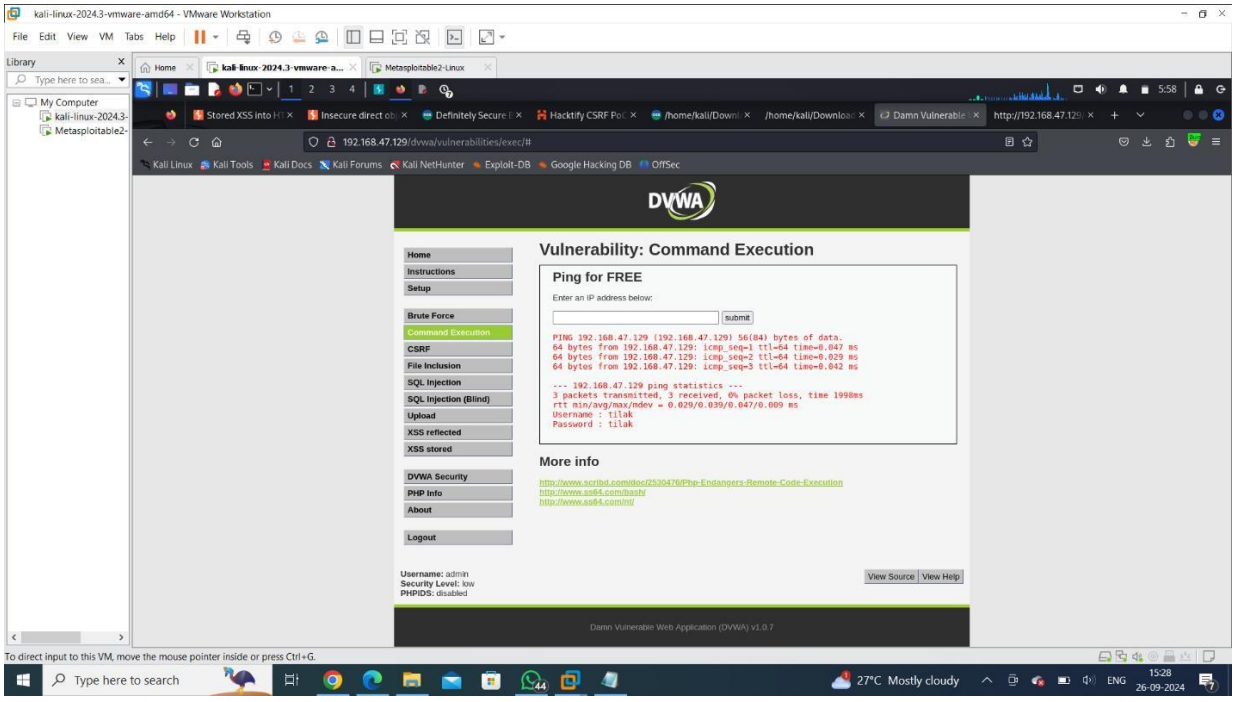
1. Open DVWA and log in using the default credentials (`admin` / `password`).

2. Go to the "Command Execution" section.

3. Enter the IP address of the target (e.g., your Metasploitable 2 VM).

4. Optionally, attempt to exploit the vulnerability by injecting additional commands using operators like `;` or `&&` (e.g., `192.168.47.129; cat tilak.txt).

5. Submit the request and check the output for any command injection results.



Stored Cross-Site Scripting (XSS)

-The Stored Cross-Site Scripting (XSS) vulnerability in DVWA.

- Entered a name (`test`) and a script payload (`<script>alert("Hi")</script>`).

- When the page is reloaded, the script executes and triggers an alert with the message "Hi", demonstrating how the input is stored and executed when accessed later.

Steps to replicate:

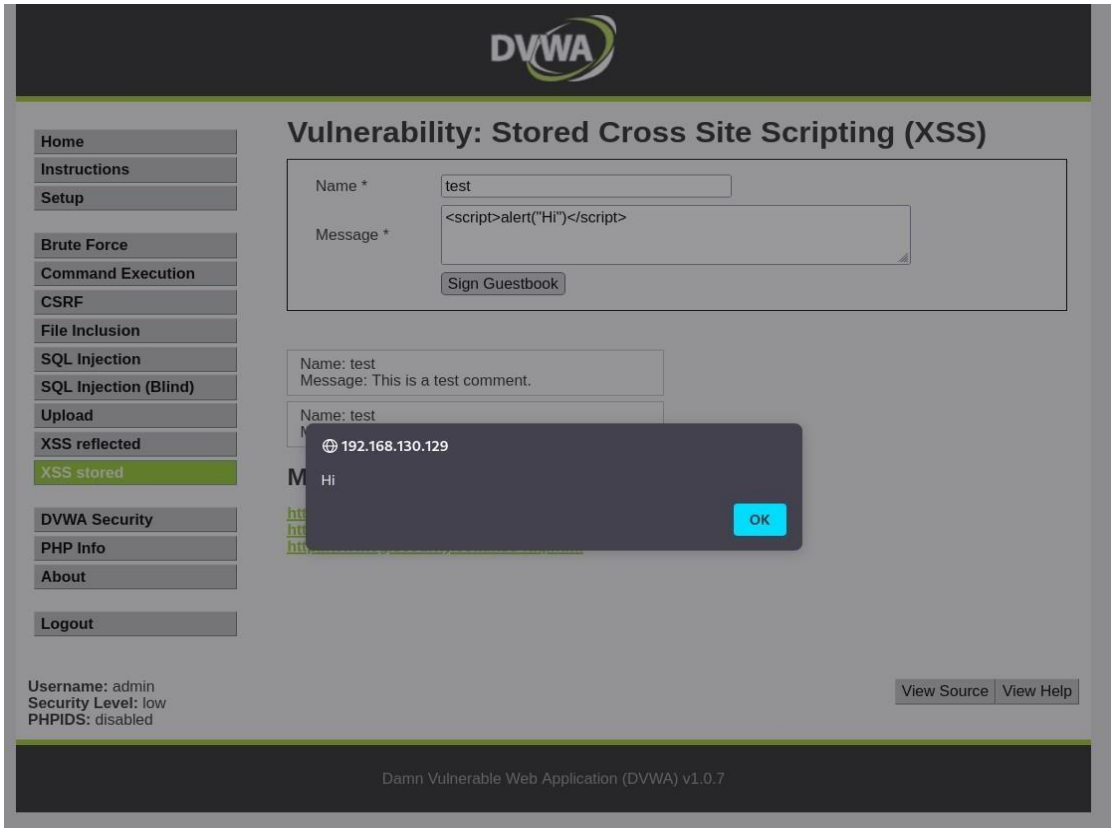
1. Go to the "XSS Stored" vulnerability section in DVWA.

2. Enter a name (e.g., "test") and a script (e.g., `<script>alert("Hi")</script>`).

3. Submit the form.

4. Reload the page or navigate to the area where the script would execute. In this case, the script is stored and executed when the page containing the "guestbook" or comment section is revisited.

5. You should see a pop-up alert message confirming the XSS attack.



DVWA Security Settings

- This screenshot displays the DVWA Security settings page.

- The current security level is set to "high", which would harden DVWA's vulnerability points, making exploitation more difficult.

- The screenshot also mentions PHPIDS (PHP-Intrusion Detection System), which can be enabled to detect and block malicious activity in real-time.

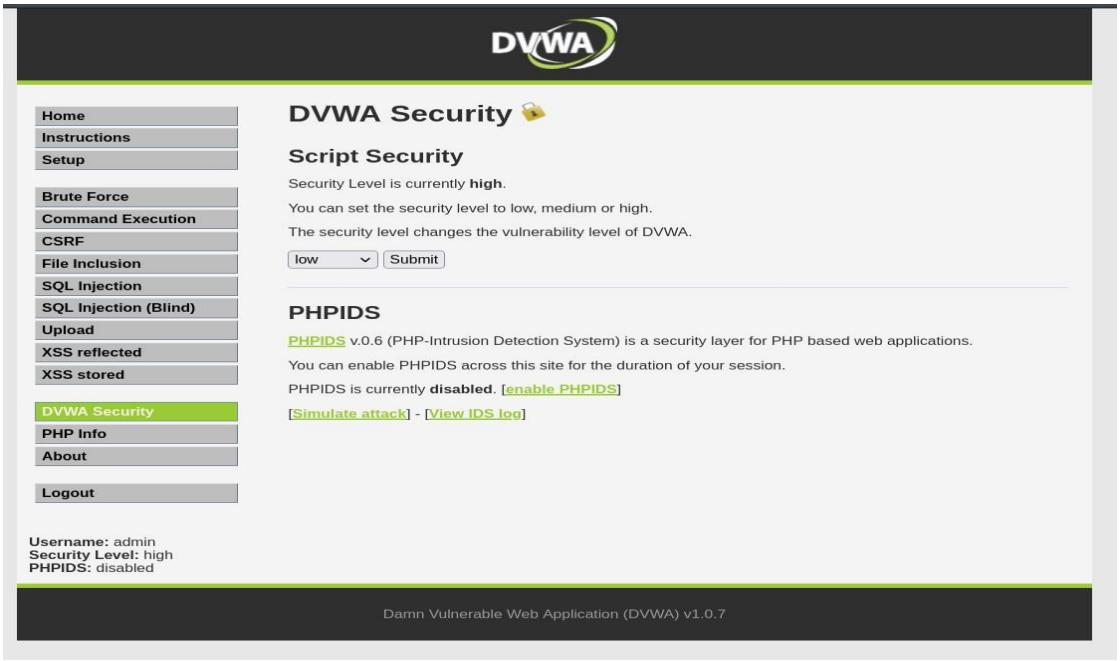
Steps to replicate:

1. Go to the "DVWA Security" section.

2. Set the security level to "low", "medium", or "high" depending on how difficult you want the challenges to be. For the simplest attacks, set the security level to "low".

3. Enable PHPIDS if you want DVWA to actively detect your attacks. This is helpful for understanding how intrusion detection systems work in real web environments.

4. Simulate different attacks from the DVWA sections (such as Command Execution or XSS) and observe how PHPIDS reacts.



Reflected Cross-Site Scripting (XSS Reflected)

Overview:

- Reflected XSS occurs when input data is immediately returned to the user without proper sanitization, leading to malicious scripts being executed in the victim's browser.

Steps to replicate:

1. Go to the XSS Reflected section in DVWA.

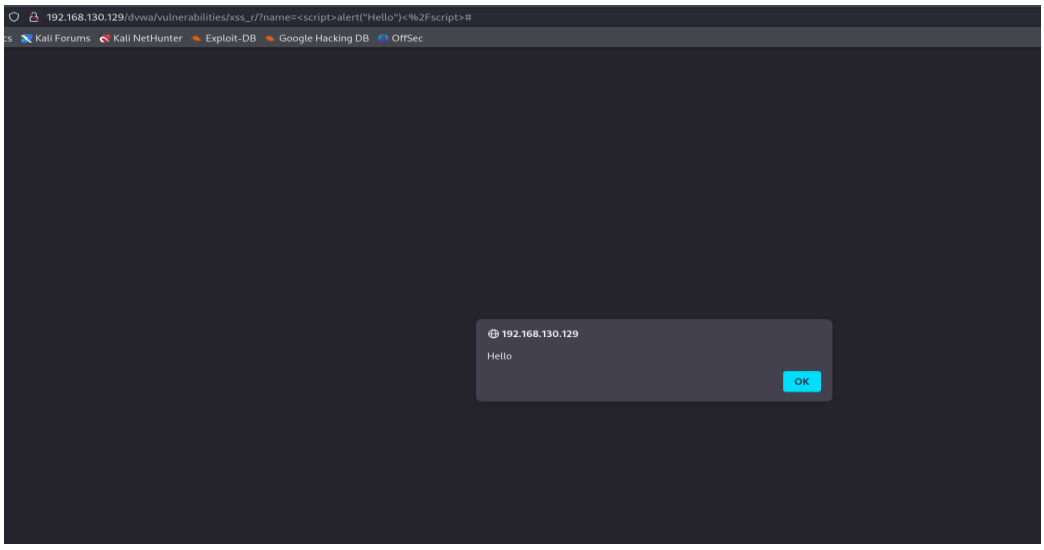
2. Enter a payload like `<script>alert('XSS')</script>` into the form field.

3. Submit the form, and the script should be executed in the browser.

4. Check the URL for parameters with the payload, which can be used to share and exploit the vulnerability.

192.168.47.129; cat /home/msfadmin/tilak.txt

- Pop-up confirmation means the site is vulnerable.



7) File path traversal, simple case

Objective:

To identify and exploit File Path Traversal vulnerabilities, where an attacker manipulates file paths to access unauthorized files on the server.

- **Vulnerability Name:** File Path Traversal (Simple Case)
- **IP Address (Vulnerable Machine):** *192.168.47.129*
- **Severity of Vulnerability:** Critical
- **Impacts:** Unauthorized access to files on the server, potentially leading to information disclosure

Steps:

1. Understand Path Traversal:

- Occurs when user input is used to construct file paths without proper validation, allowing attackers to navigate directories and access files outside the intended location.

2. Identify Vulnerable Parameters:

- Look for parameters in URLs or form inputs that handle file paths (e.g., `?file=report.txt`).

3. Test for Vulnerabilities:

- Manipulate the file path by using traversal sequences like `../` to attempt accessing sensitive files.

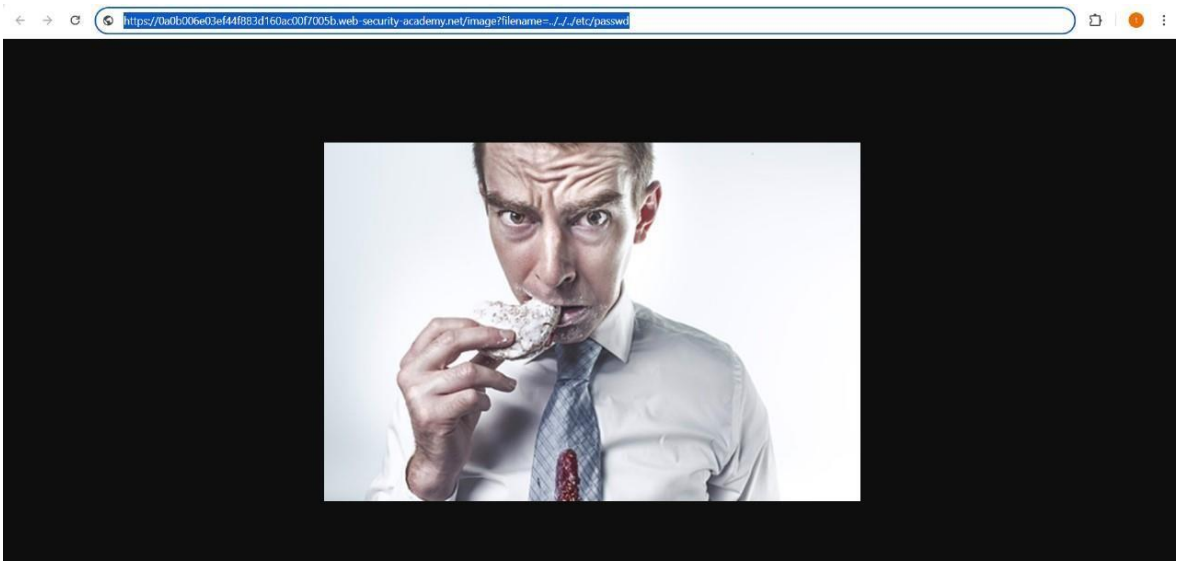
- Example:

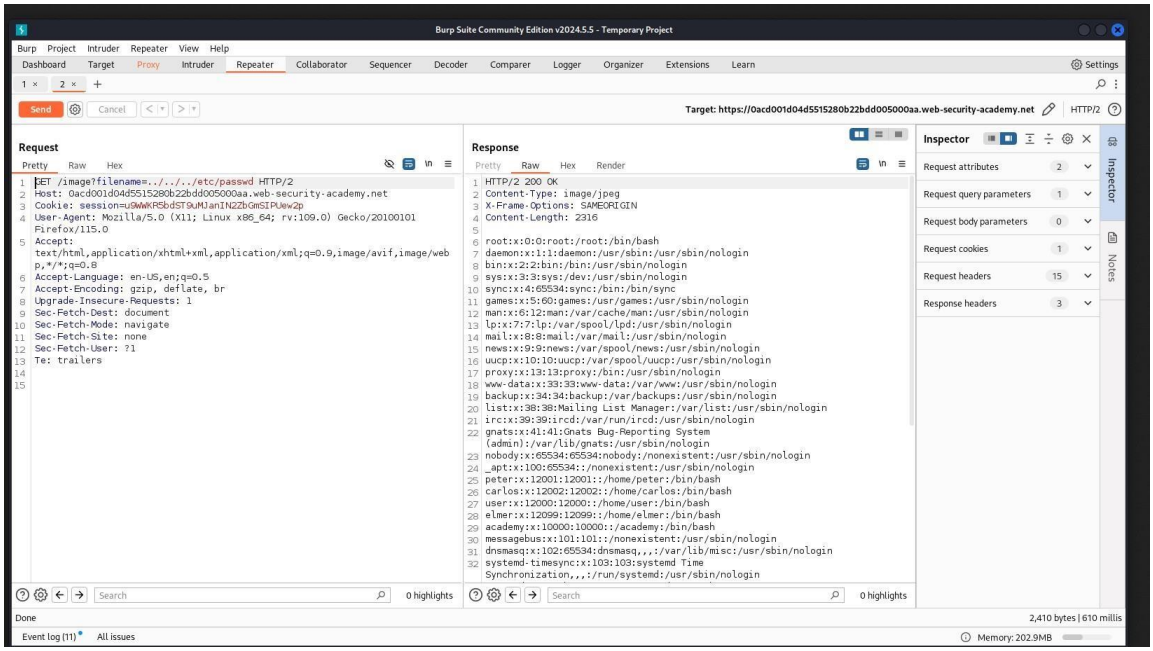
https://example.com/?file=../../etc/passwd

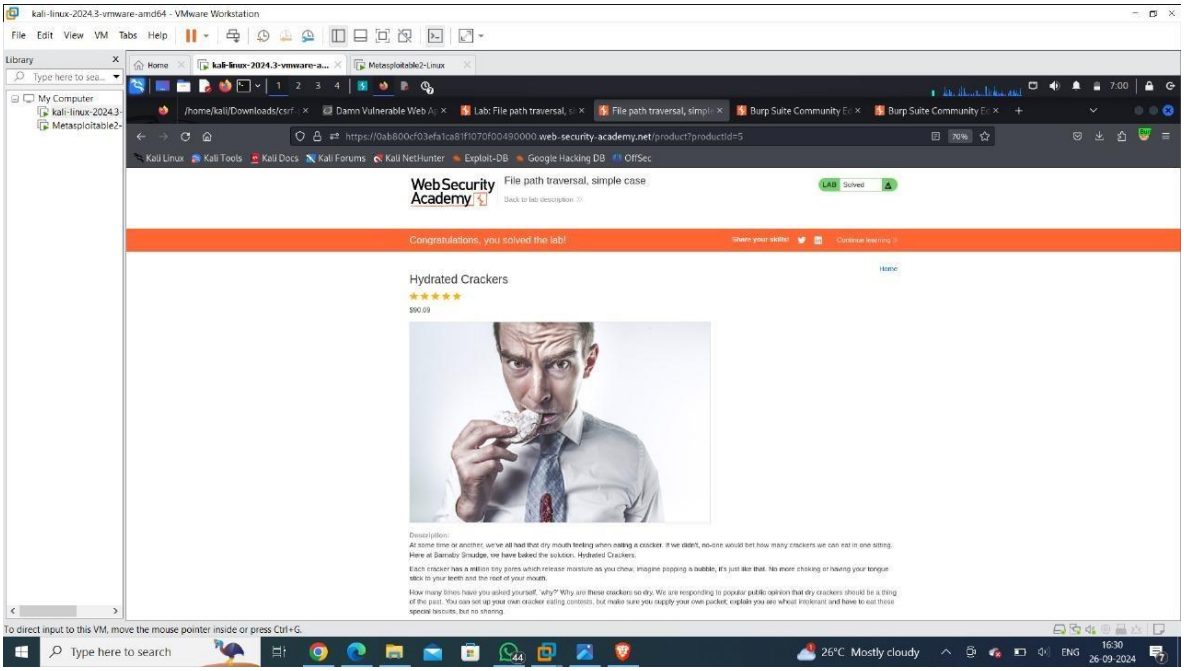
- If the server returns the contents of the file, the vulnerability is confirmed.

4. Remediation:

- Recommend using whitelisting for valid file paths and restricting access to directories outside the intended scope.






kali-linux-2024.3-vmware-amd64 - VMware Workstation
File Edit View VM Tabs Help
Library
My Computer
kali-linux-2024.3-
Metasploitable2-
Home
kali-linux-2024.3-vmware-a...
Metasploitable2-Linux
/home/kali/Downloads/csrf...
Damn Vulnerable Web Ap...
Lab: File path traversal, s...
File path traversal, simple...
Burp Suite Community Ed...
Burp Suite Community Ed...
https://0ab800cf03efa1ca81f1070f00490000.web-security-academy.net/product?productId=5
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Web Security Academy
File path traversal, simple case
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home
Hydrated Crackers
$90.09
Description:
At some time or another, we've all had that dry mouth feeling when eating a cracker. If we didn't, no-one would bet how many crackers we can eat in one sitting.
Here at Barnaby Smudge, we have baked the solution. Hydrated Crackers.
Each cracker has a million tiny pores which release moisture as you chew, imagine popping a bubble, it's just like that. No more choking or having your tongue stick to your teeth and the roof of your mouth.
How many times have you asked yourself, 'why?' Why are these crackers so dry. We are responding to popular public opinion that dry crackers should be a thing of the past. You can set up your own cracker eating contests, but make sure you supply your own packet, explain you are wheat intolerant and have to eat these special biscuits, but no sharing.
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
26°C Mostly cloudy
ENG
16:30
26-09-2024