**Assignment 4**

# Finding, Exploiting, and Fixing Vulnerabilities in Web Apps

(Nishant Chaudhary)

Get the IP address:



**Vulnerabilities:**

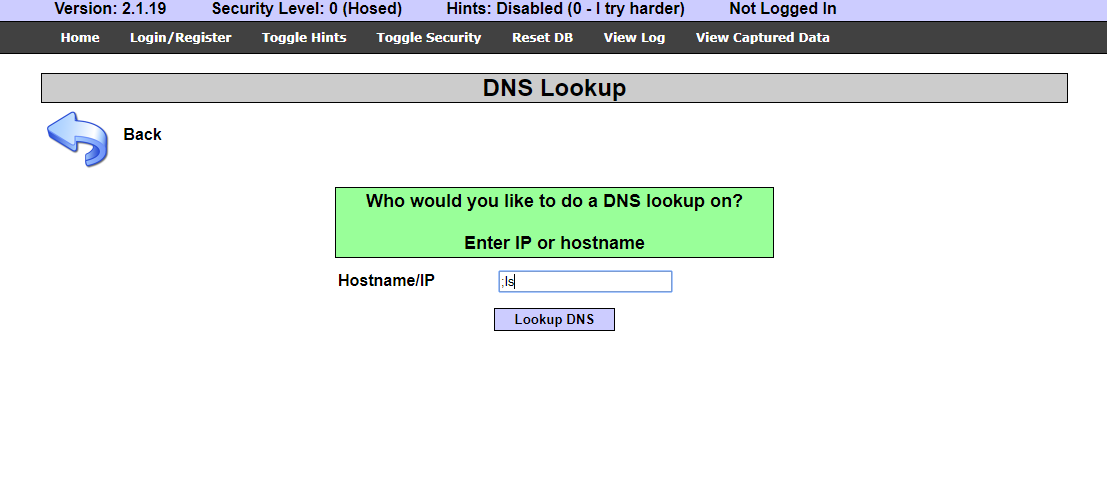
1. **A1 Injection - Command injection:**

Redirect the DNS lookup script to separate directory as shown below:



Open DNS lookup page from browser using the following url:

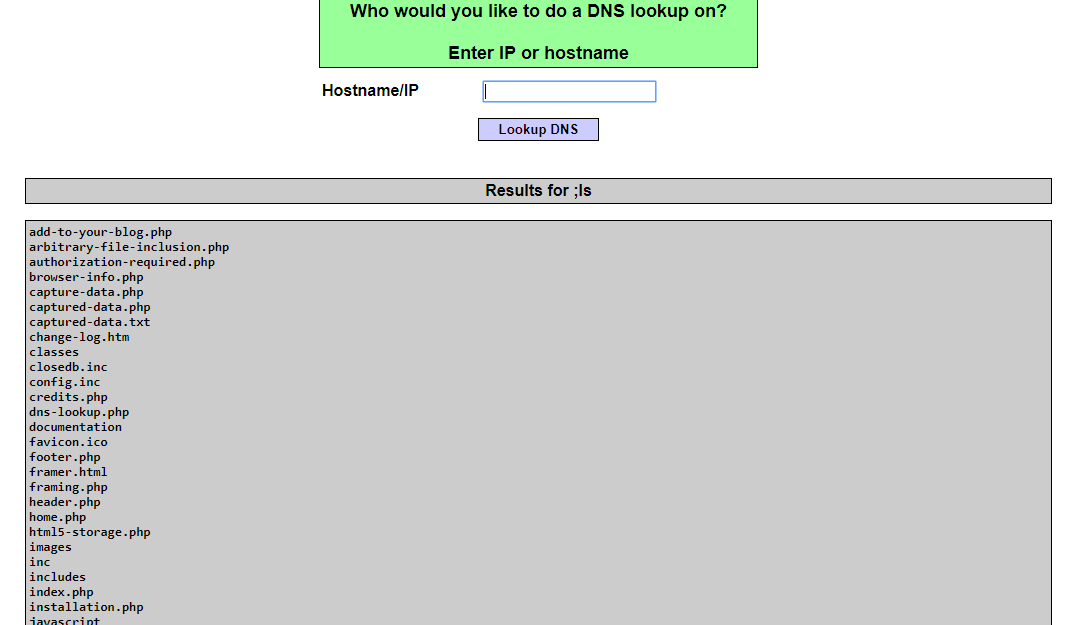
http://11.121.139.26/mutillidae/index.php?page=dns-lookup.php



When used ‘;’ as starting letter, it ends the before script and executes whatever command provided later as shown above.

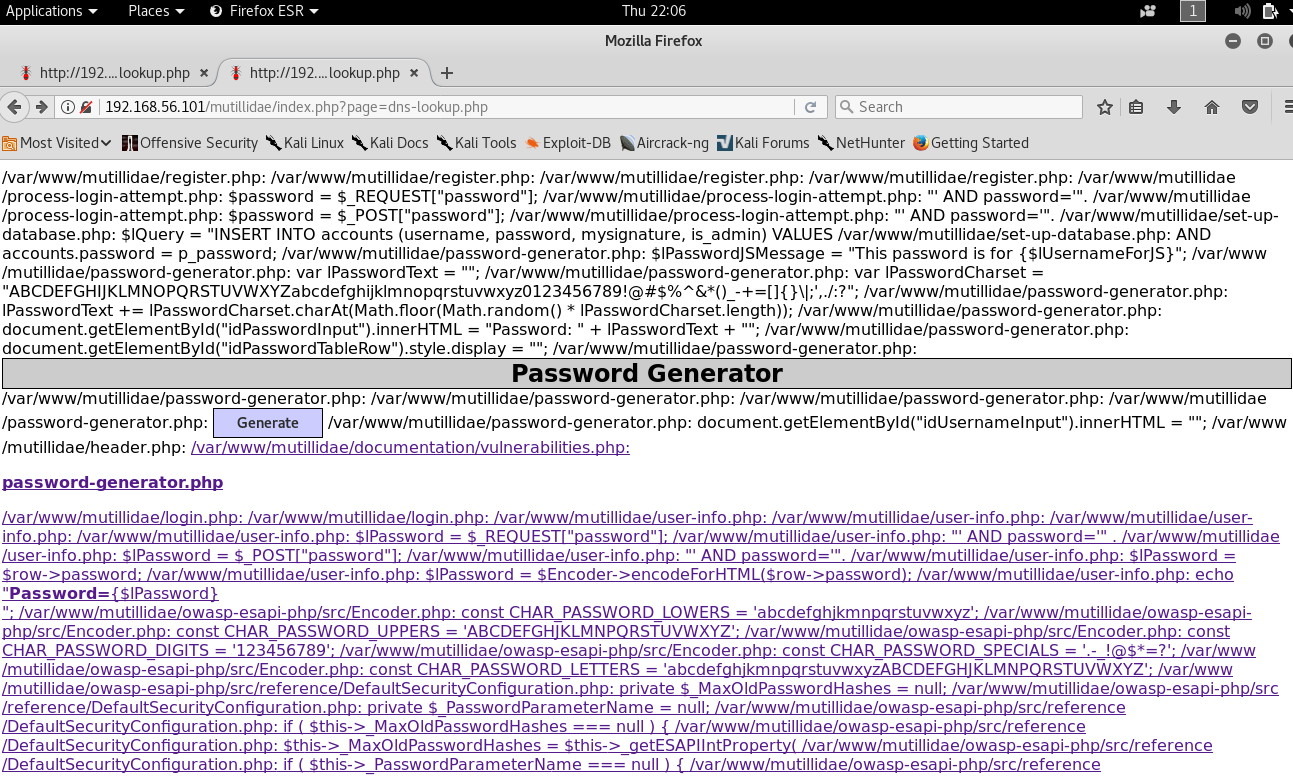
First, find the directory in which the scripts are located as shown below.

Type the below command to inspect and find username and password as shown below.



Find the password string in the results using the command as shown below.

; find /var/www/mutillidae -name "\*.php" | xargs grep -i "password" | grep "="



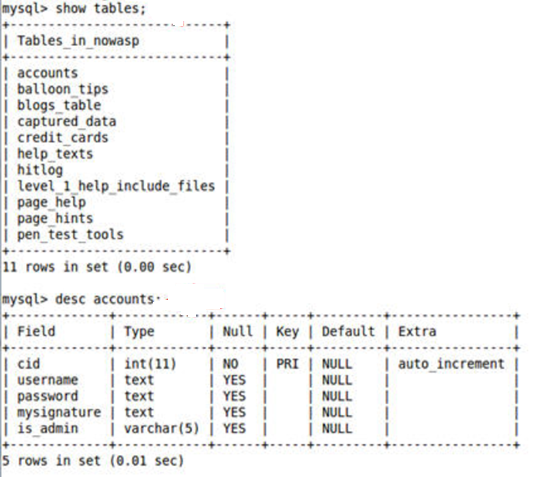
Find the username using the command below.

; find /var/www/mutillidae -name "MySQLHandler.php" | xargs egrep -i '(user|login)' | grep "="

Using the details found, login to the mySQL using the below command from kali linux as shown below.



After successful exploit, the following commands can be executed into metasploitable 2 os.



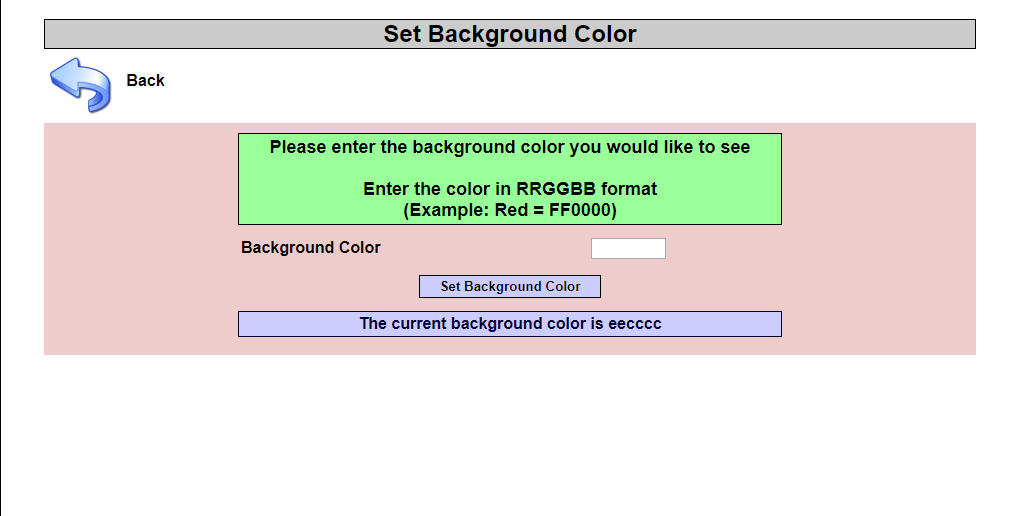
**Solution:**

To fix this issue, pass the values to variables first and then perform input validations so that input in a specific format, instead of directly passing values to the function, in this case directly to a database query.

This will prevent command injection to a certain extent.

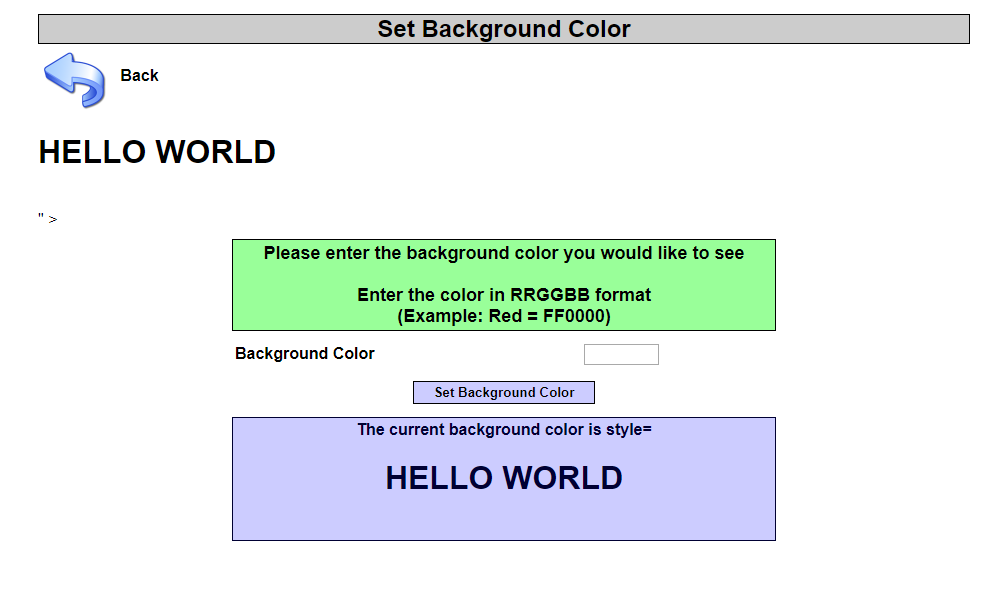
1. **A2. Cross Site Scripting(XSS) Reflected attack:**

The exploit is performed by using color checkbox as shown below:



The text is used to execute a script which injects a H1 tag into the webpage. The below CSS script is executed in the text box as shown below.

style=<body color:#"**"><H1>HELLO WORLD</H1><br anything="**">

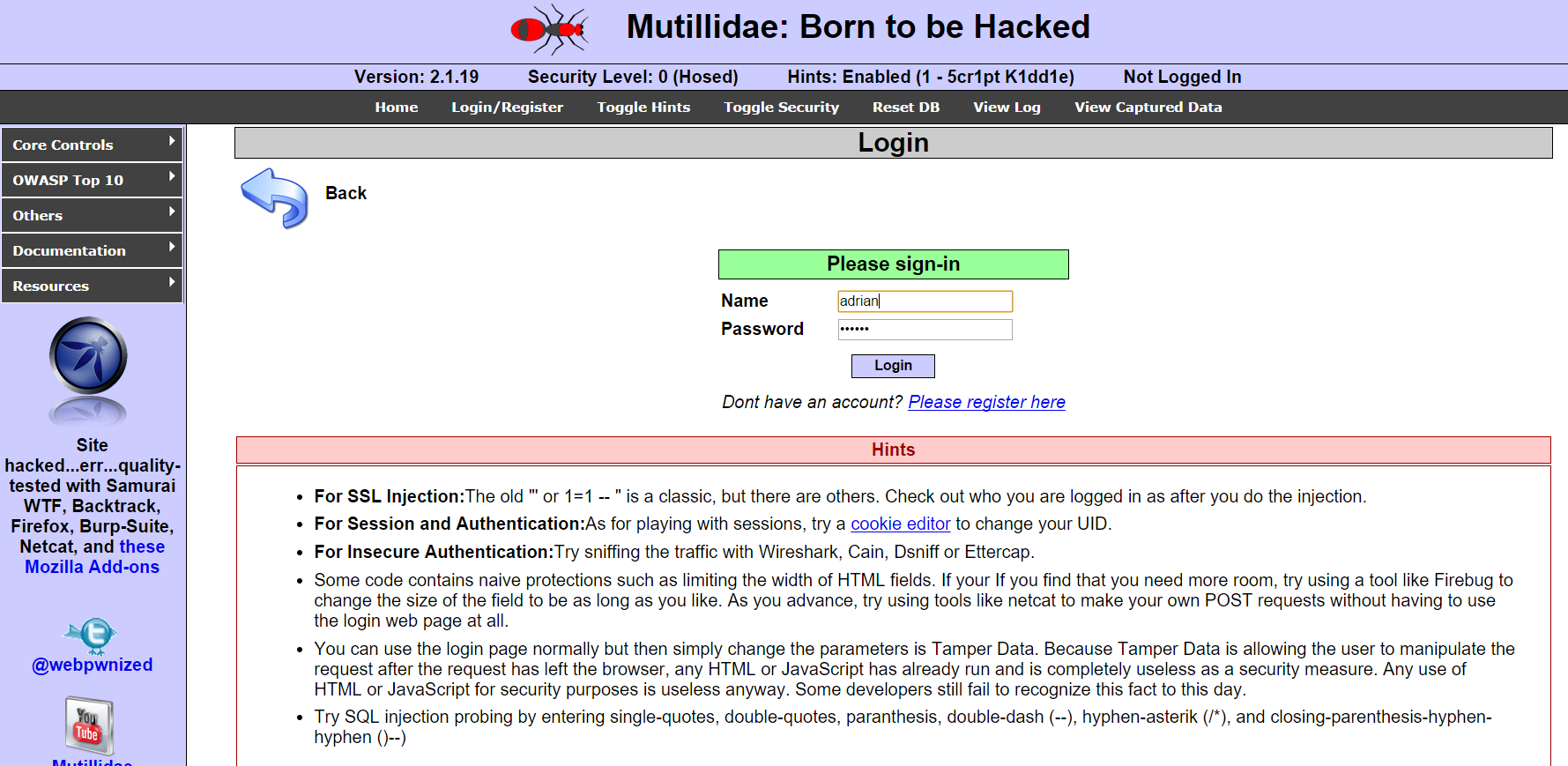


**Solutions:**

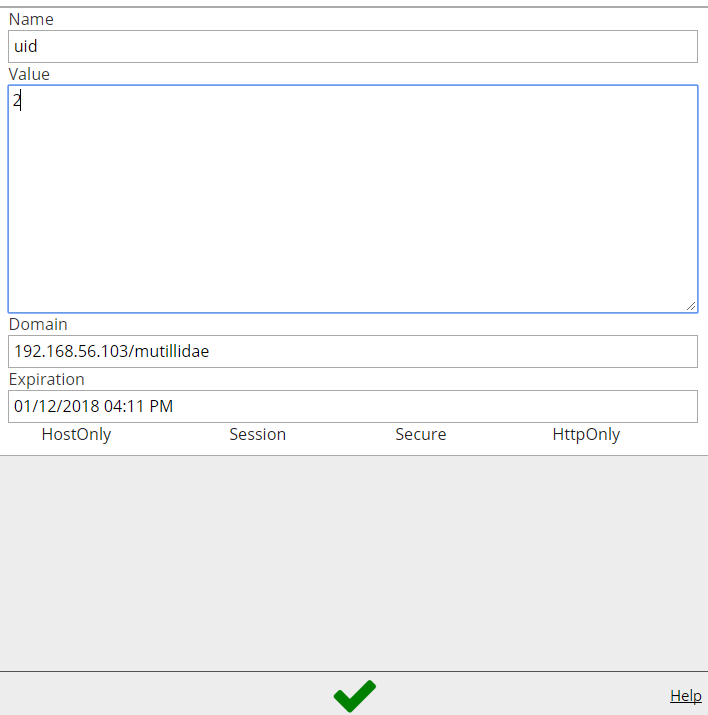
The viable solutions for this attack are described below.

1. Use simple data filters to filter to prevent such attacks.
2. Always prefer to use existing functions provided by the language. In php use strip\_tags(), utf8\_decode() etc. to perform filter.
3. Always assume data is invalid and trust it only when it is proven as valid by one of the system functions.
4. Initialize data to some default values and when some invalid is found, redirect the code to the default values and prevent attacker to inject the values.
5. **A3. Broken Authentication and Session Management:**

Open the following webpage and login to perform the exploit



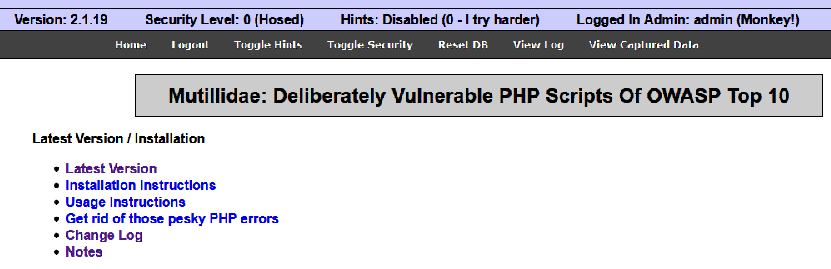
The exploit uses a cookie manager to edit cookies and redirect login. Edit and perform the exploit as shown below.



Edit value to 1 and save the cookie and reload page.



After reloading, the user will login as shown below.



**Solution:**

Attacker uses leaks and flaws in the authentication or session management functions to impersonate the users.

**Password Strength** – Have restrictions based on a minimum size and complexity.

**Password Use** – Have restrictions based on number of login attempts per unit time and log number of failed attempts.

**Password Change Controls** – A single password change mechanism should be used wherever users can change a password, regardless of the situation. Users should always be required to provide both their old and new password when changing their password.

**Password storage** – All passwords have to be stored in hashed or encrypted forms.

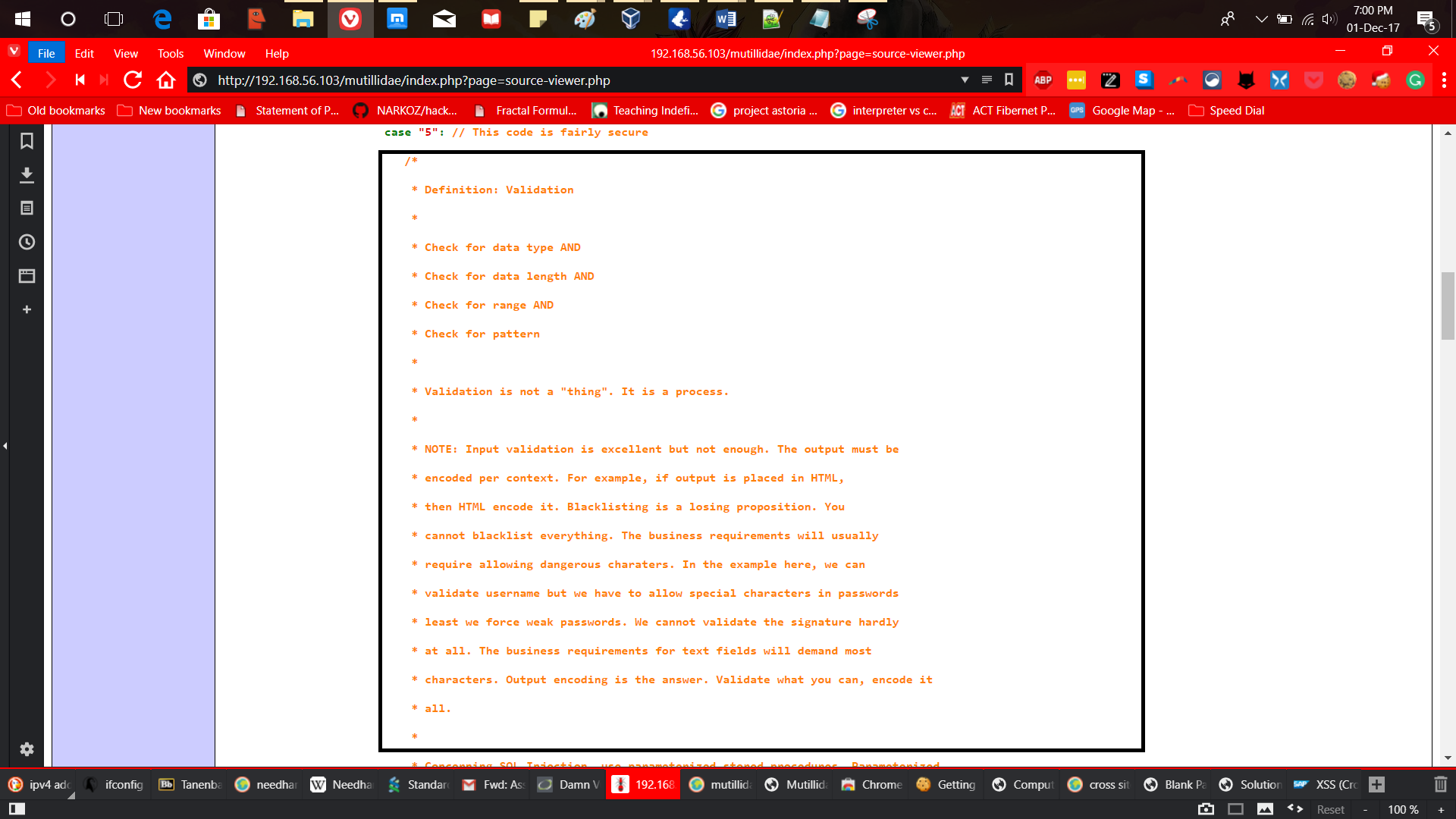
**Session ID protection** – Select the session via SSL.

**Browser Caching** – Authentication and Browser data should never be submitted as a part of GET. POST method should be used.

1. **A4. Insecure Direct Object References**

The webpage shown below gives more information about the application. Using this the exploiter can gain more information about the website and use the knowledge to exploit more.

The webpage shown below gives more information about the application. Using this the exploiter can gain more information about the website and use the knowledge to exploit more.



The attacker can gain information about server configuration and how the data is organized as shown in the below.



**Solution:**

The attack uses comments to gain more insight about the application. The probable solution is to refrain from mentioning server credentials, server configuration and input validation information on the website.

1. **A5. Cross Site Request Forgery (CSRF)**

To perform the exploit, open the DVMA website and login with the below credentials.

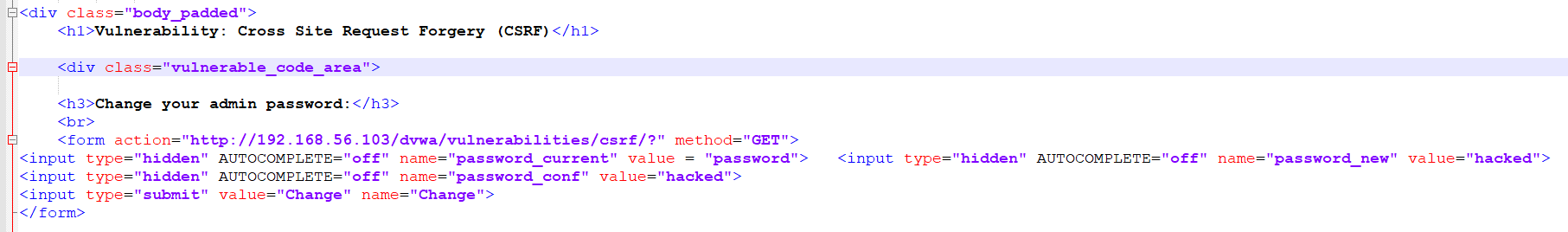
Username: admin

Password: password

The exploit is performed as shown below.



Change the form as shown below.



Save the edited page as html and open it in the browser. Then click on ‘change password’ to change password as shown below.



The below message is shown that the password is changed.



**Solutions:**

Occurs when an unwanted action on a trusted site for which the user is authenticated, is performed.

S**ynchronizer Tokens** – It is unique per sessions and has large random values. Generated by a cryptographically secure random number generator.

**Double Cookie Defense** – Send a random value in both a cookie and as a request parameter. Have the server verify if the cookie and the request value is the same.

**Encrypted token pattern** – After authentication, have the server generate a unique. Token comprised of the user’s ID, a timestamp values and a cryptographic nonce, using a unique key available only to the server.

**Custom Request Header** – A non-complex defense suited for AJAX endpoints. It relies on same origin policy (SOP) restrictions that only JavaScript can ve used to add a custom header and only within its origin. By default, browsers do not allow JavaScript to perform Cross-origin requests.

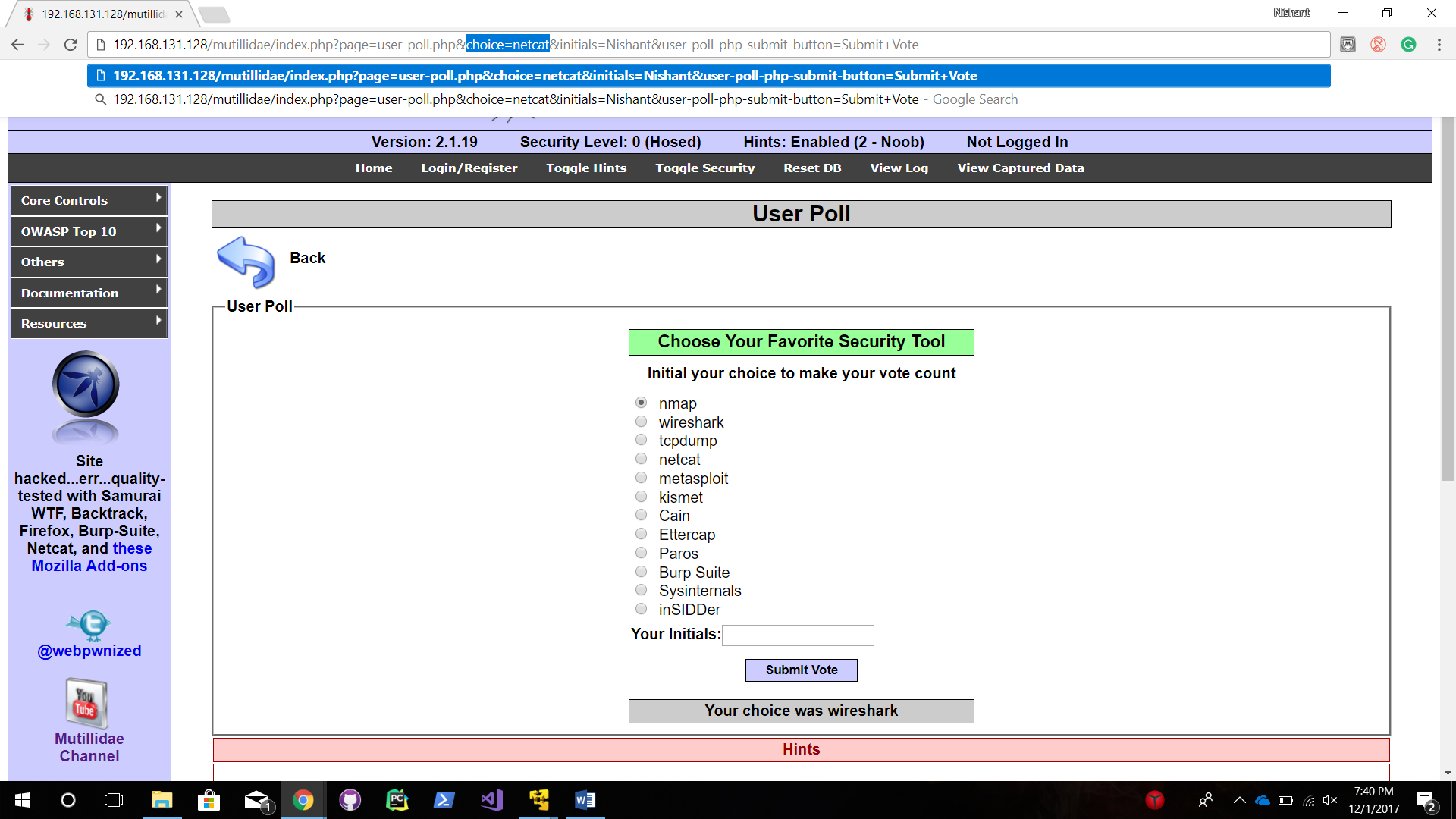
1. **A6: Security Misconfiguration – GET for POST:**

The exploit allows user to edit URL and proceed to different selections as shown below.

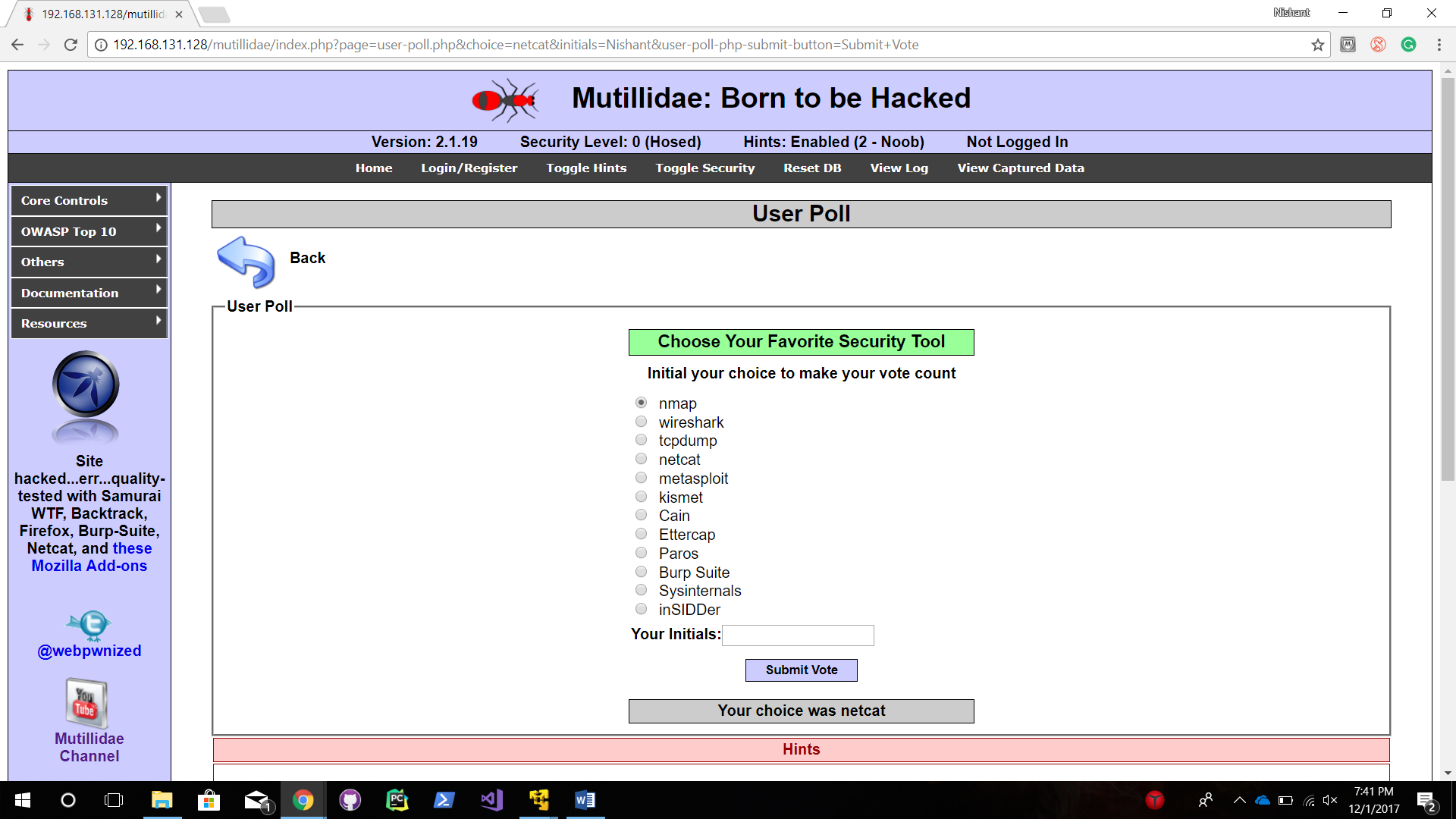
The exploit allows user to edit URL and proceed to different selections as shown below.



The exploit is performed by editing the URL as shown below.



After editing the URL, the webpage shows a message that another option is selected as shown below.



**Solution:**

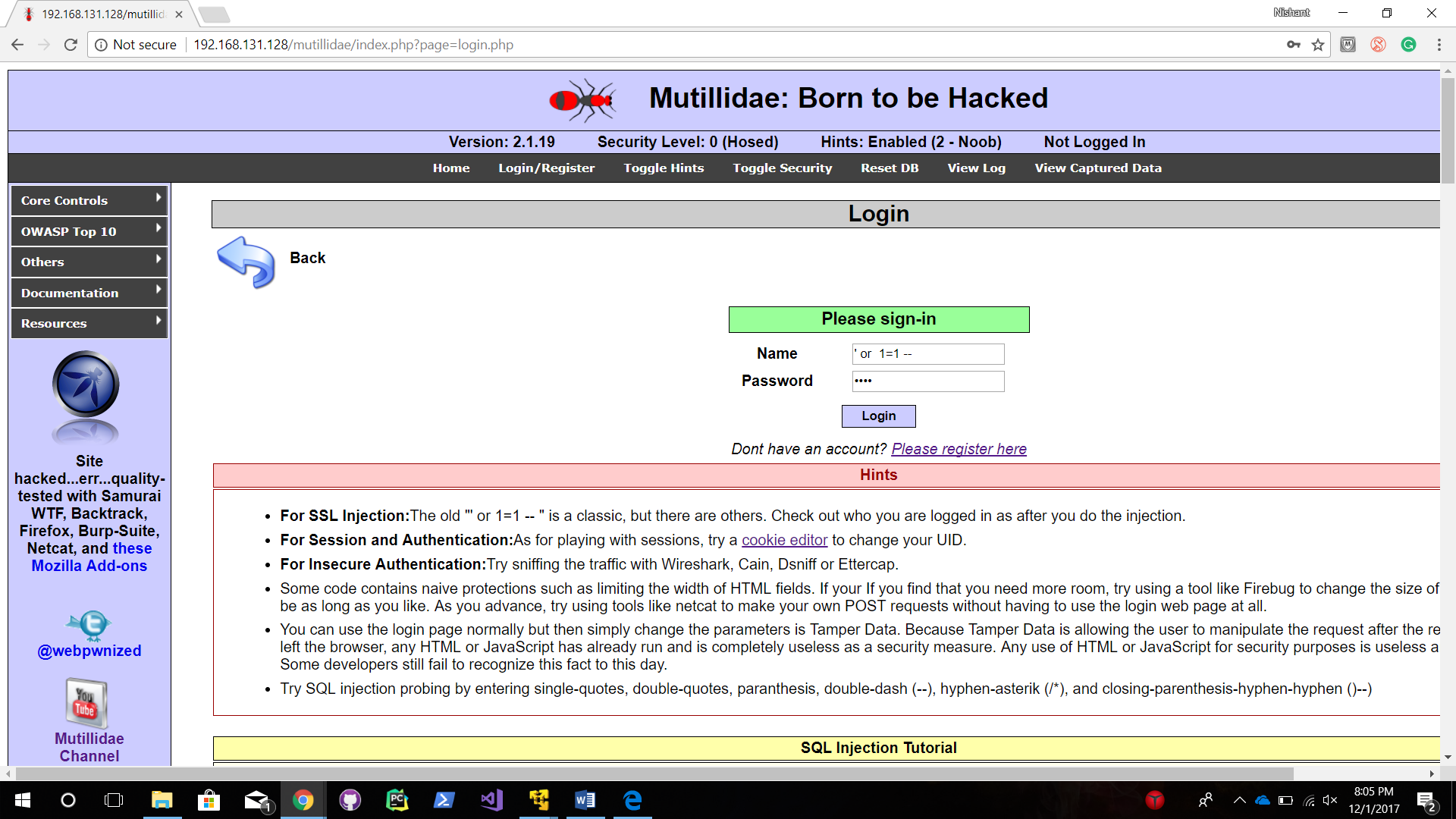
The attack is performed by leveraging the Http GET method. The solution for the vulnerability is to use the Http POST method instead of GET.

1. **A1. SQL Injection – Bypassing login bypassing:**

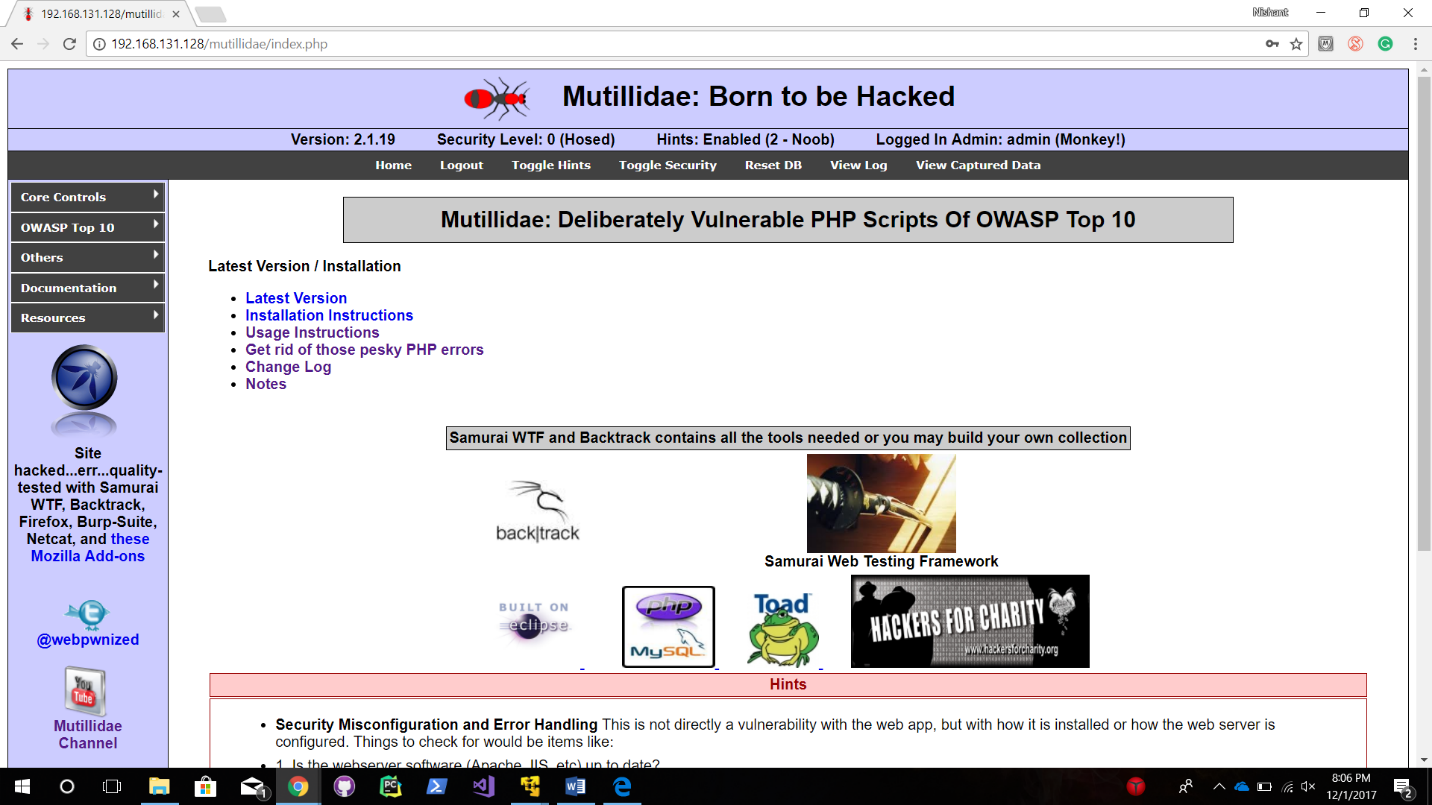
The attack enables exploiter to perform a login bypass by using the following code.

"' or 1=1 -- "

The exploit is performed as shown below.



The webpage redirects and opens ‘admin’ portal as shown below.



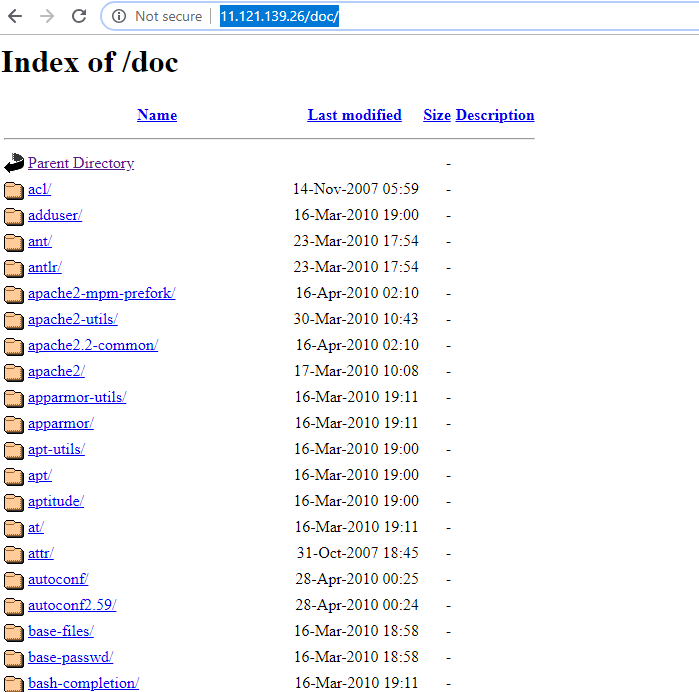
1. **A6. Security misconfiguration: Browsable directories:**

The exploit is performed as described below.

Browse the below URL from web browser.

<http://11.121.139.26/doc/>

This exploit enables attacker to browse server directories by entering the below URL directly from the browser without proper authorization and authentication as shown below.



1. **A7. Insecure Cryptographic Storage:**

The exploit is shown below.

Open the SQL injection page to perform attack as shown below.



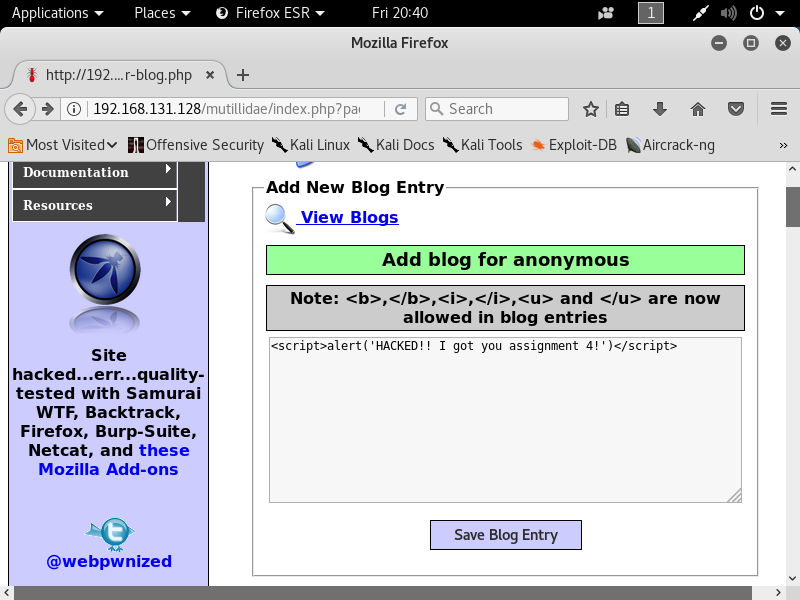
To exploit enter the below command, it will display the list of usernames and details directly on the screen.

' union select @@version, null—

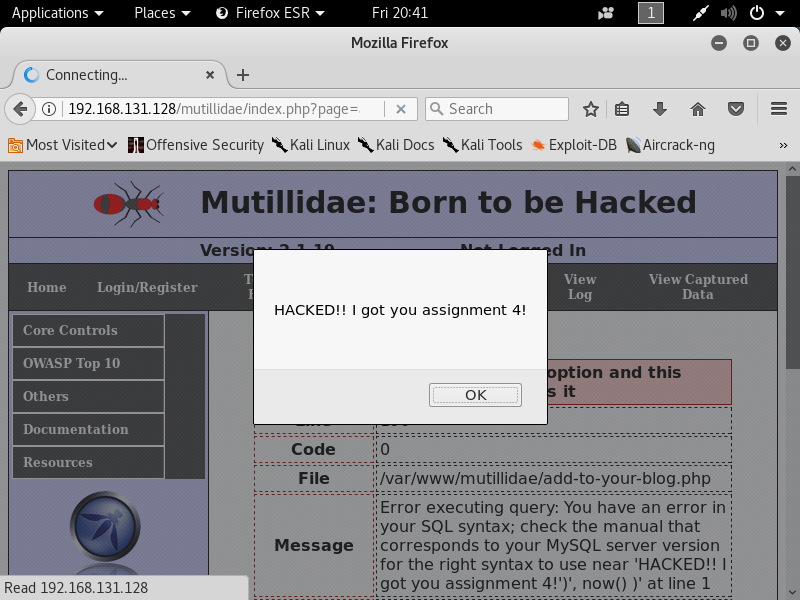


1. **A2: Cross Site Scripting (XSS) – Persistent attack:**

The exploit is performed as shown below.



Enter a javascript code snippet in the entry as shown above. This will execute the code and show a pop-up as shown below.



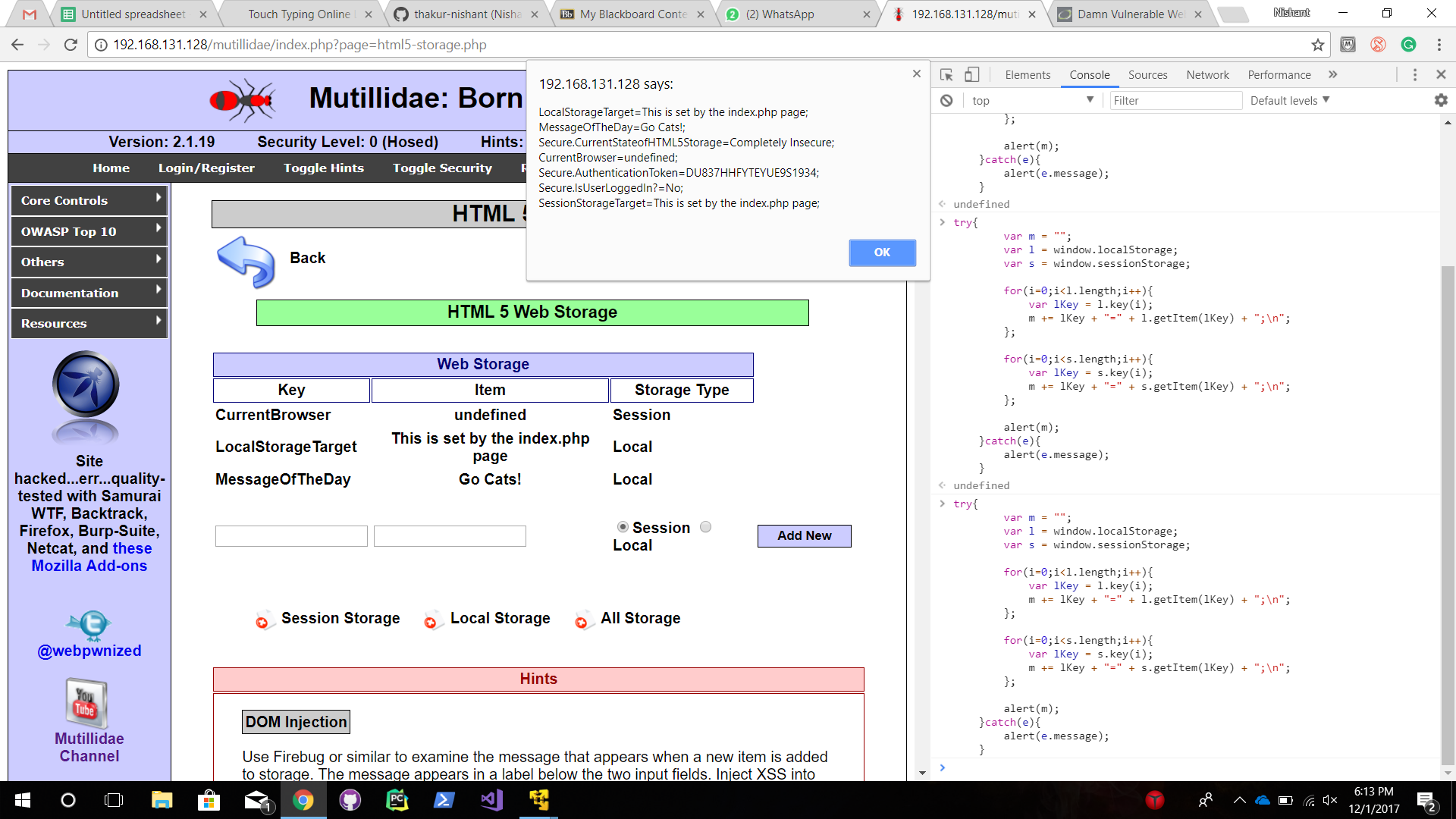
**BONUS:**

1. **A7. Insecure Cryptographic Storage:**

The exploit uses the vulnerability that the storage and session information is included in the HTML webpage.

The exploit is performed as shown below.

By executing the JavaScript code in the console of the web browser the attacker is able to get access to the session information, secret authentication token information for encryption as shown below.



**Solution:**

This attack occurs when sensitive data is not stored securely.

- Make sure offsite backups are encrypted, but the keys are managed and backed up separately.

- Ensure appropriate strong standard algorithms and strong keys are used, and key management is in place.

- Ensure passwords are hashed with a strong standard algorithm and an appropriate salt is used.

- Ensure all keys and passwords are protected from unauthorized access.

1. **A8. Failure to Redirect URL Access:**

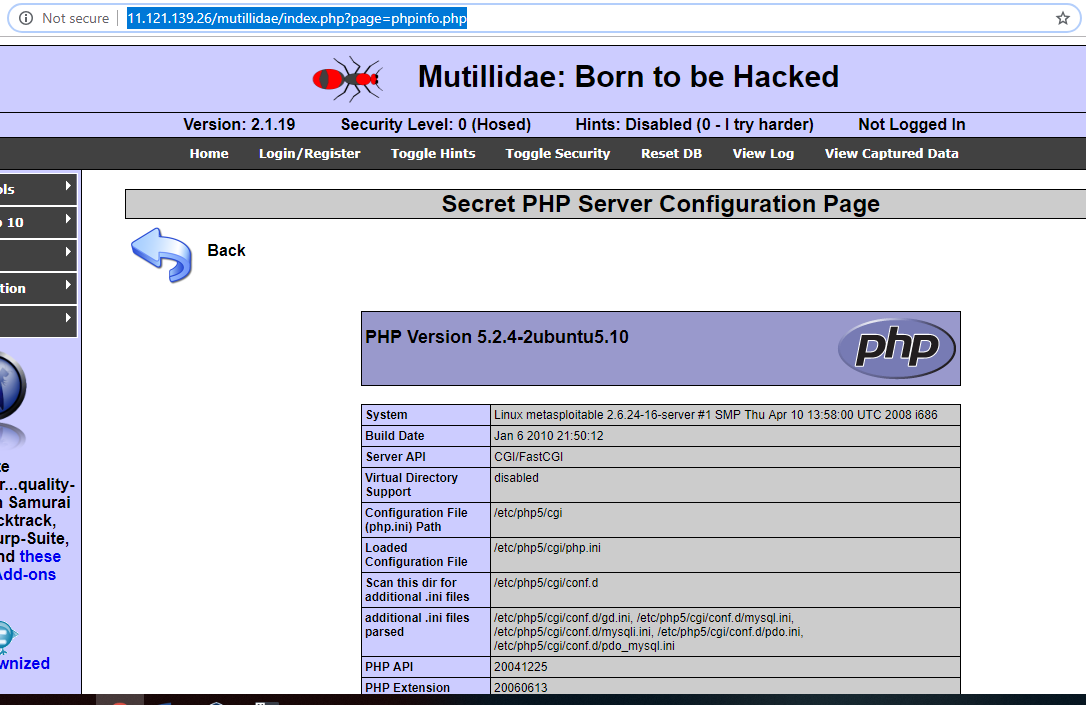
The exploit allows attacker to access unauthorized URLs by changing URL.

The exploit is performed as shown below.

Open the below webpage and edit URL as shown below.



The webpage redirects to an unauthorized administrator page as shown below.



**Solution:**

Developers attempt to hide functionality from users by creating hidden pages. This is an example of the failure to Restrict URL access vulnerability.

- First select an approach that requires proper authentication and authorization for every page.

- These authorization and authentication policies must be role based to minimize the effort required to maintain the policies.

- The policies are required to be highly configurable, in order to minimize any hard-coded aspects of the policy.

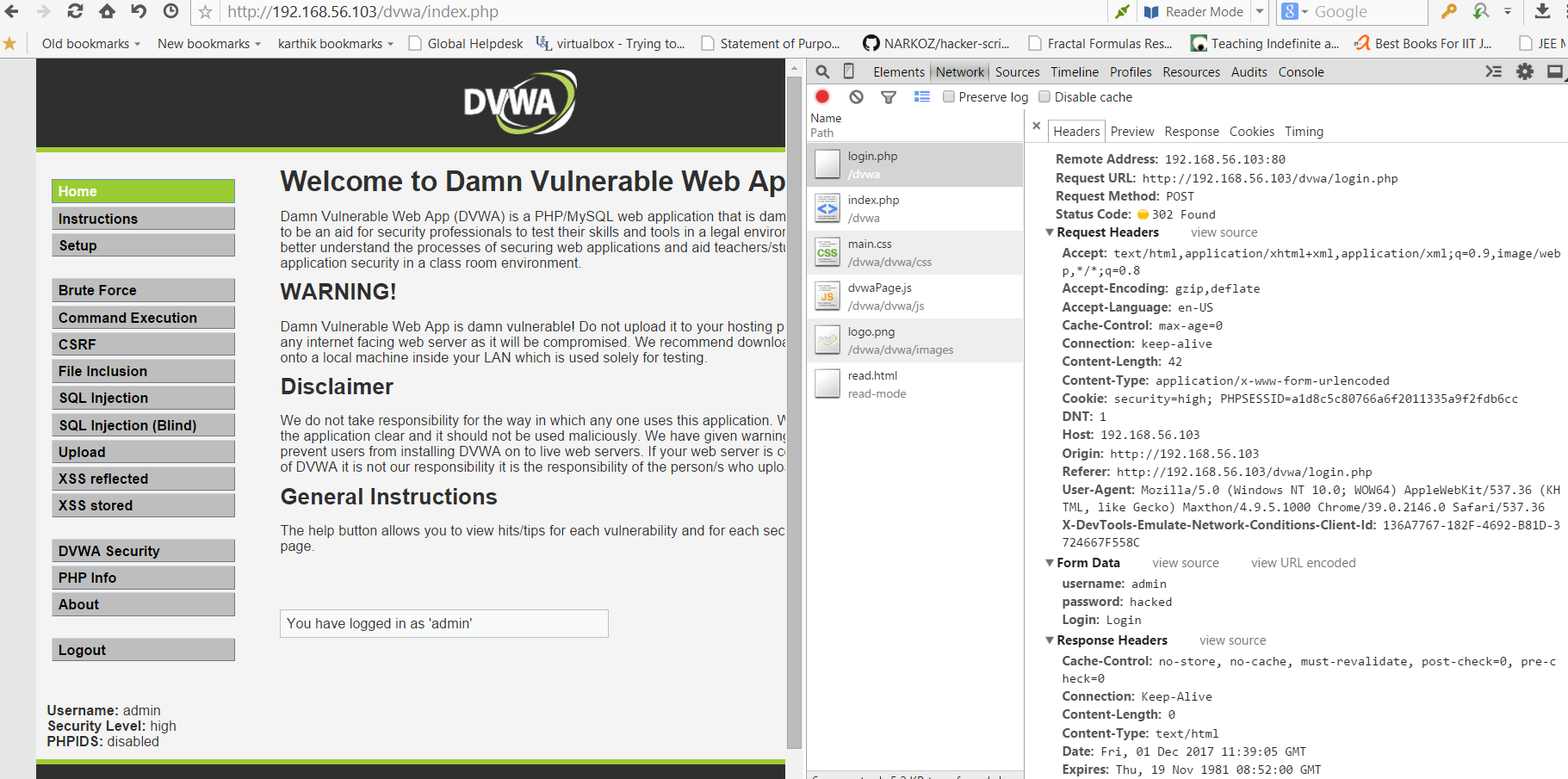
- The enforcement mechanisms should deny all access by default, requiring explicit grants to specific users and roles for access to every page.

- If the page is involved in a workflow, check to make sure the conditions are in the proper state to allow access.

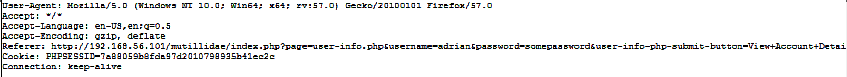
1. **A9. Insufficient Transport Layer Protection:**

The exploit is executed as shown below.

Login to DVMA website and run a network scan using the inspect element. The username and password can be seen in the header as shown below.



The username and password can be viewed by hacker as shown below.



**Solutions:**

Traffic flowing between two points is not always secure, and thus an attacker can easily perform man-in- the-middle attacks.

**Browser Solution:** Implement HTTP Strict Transport Security in the browsers to enforce secure connections. Also implement Certificate and Public Key pinning in browsers where applicable.

**Perimeter Solution:**

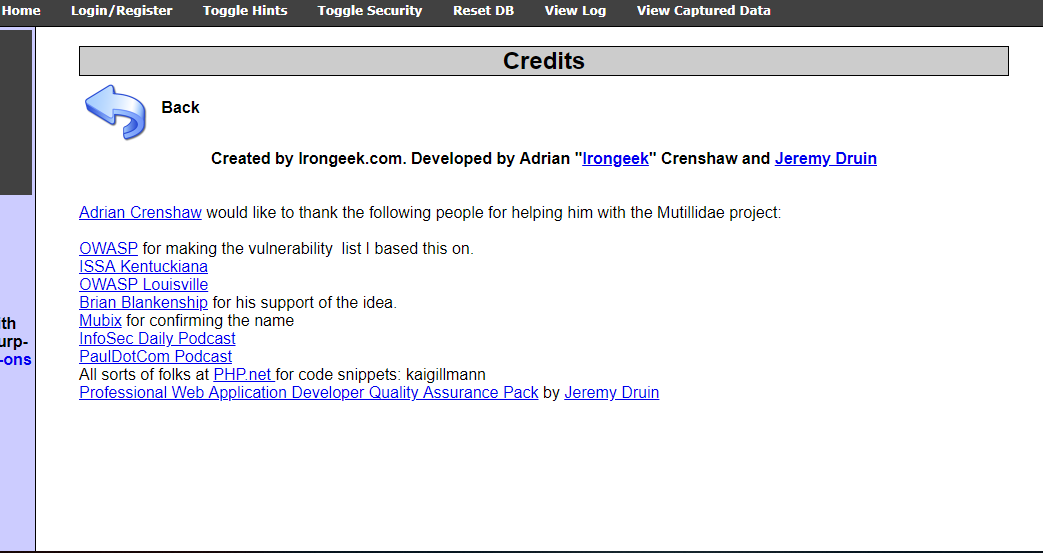
Enforce HTTP Strict Transport Security

* Redirect all HTTP requests to HTTPS
* Configure SSL properly on the server.
  + Disable weak SSL/TLS protocols
  + Disable weak ‘export’ algorithms
  + Use SSL certificate with a minimum key size of 2048 bit
* Make sure session key size is 128 bits
* Do not use RC4 algorithm.
* Do not use MD5 hash.
* Disable Anonymous Key Establishment Algorithm
* Disable algorithms offering no encryption

1. **A10. Unvalidated Redirects and Forwards:**

Open the below page to perform the exploit.

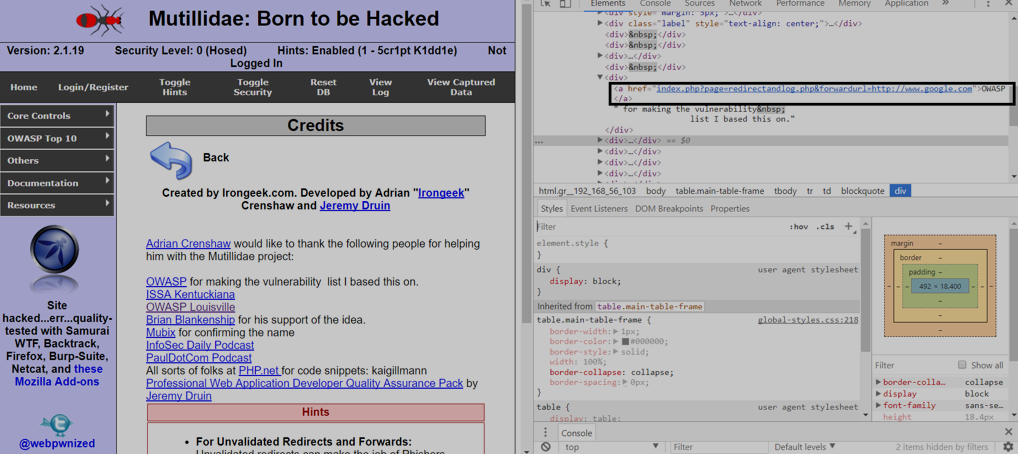
<http://11.121.139.26/mutillidae/index.php?page=credits.php>



Change the URL as shown below to redirect the page to different website.

http://11.121.139.26/mutillidae/index.php?redirectandlog.php?forwardurl=https://www.google.com

It will redirect to www.google.com as shown below.



**Solution:**

It occurs when a web application accepts untrusted input that redirects the request to a URL contained within the untrusted input.

- Validate user input and make sure it is authorized for the user.

- Do not allow URL as user input.

- Make sure that destination input is mapped to a value and the server side code translate this code to the target URL.

- Create a list of trusted URLs and verify the input against it.

- Incase redirects occur, force all the redirects to go to a different page to notify the

user of the redirect.

1. **A2: Cross Site Scripting (XSS) – Reflected attack:**

The exploit is performed as shown below.

The attack is performed by executing a javascript code as shown below:



This will execute the code and show a pop-up as shown below.

