EXPLOIT WEB APPLICATION SECURITY USING DVWA

WHAT IS DVWA?

- Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application.
- Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment,
- help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

HOW TO INSTALL DVWA?

- Go to official site: http://www.dvwa.co.uk/
- Click on download.
- Put downloaded file in localhost folder (i.e. xampp\htdocs)
- Extract the file
- Start the server
- Run the local site in browser.
- Login with below credentials
- Username: 'admin' password: 'password'

SCREENSHOT: 🐧 File Inc 🗋 CSF46: W Cross- 🔃 Data V 🛅 Plot D: 🦁 Data T 💆 Home 📓 pythor 🔀 Crop_F 🗎 my-fin 📑 Linear S CCC Q 🚾 267shi 🕭 Down 🐻 W x 🔸 ☆ @ A : ← → C ① localhost/dywa/ DVWA Welcome to Damn Vulnerable Web Application! Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment. Instructions Setup / Reset DB Brute Force The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficultly, with a simple straightforward interface. Command Injection CSRF **General Instructions** File Inclusion It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability. File Upload Insecure CAPTCHA SQL Injection (Blind) DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)! Weak Session IDs XSS (DOM) XSS (Reflected) XSS (Stored) There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue. CSP Bypass JavaScript WARNING! DVWA Security Damn Vulnerable Web Application is damn vulnerable! Do not upload it to your hosting provider's public html folder or any Internet facing servers, as they will be compromised. It is recommend using a virtual machine (such as <u>Virtual Box</u> or <u>Vihware</u>), which is set to NAT networking mode. Inside a guest machine, yor can downloading and install <u>XAMPP</u> for the web server and database. PHP Info

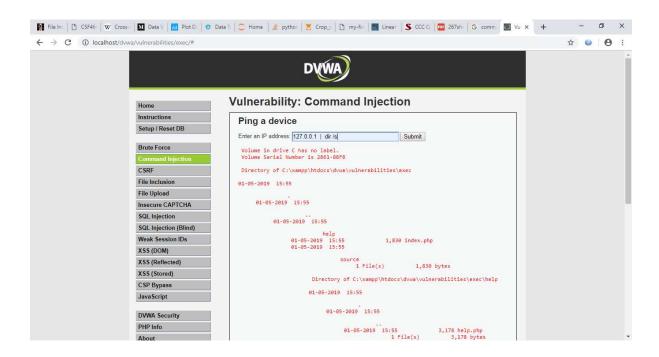
COMMAND EXECUTION/INJECTION

- DVWA 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.

PURPOSE:

- The purpose of the command injection attack is to inject and execute commands specified by the attacker in the vulnerable application.
- In situations like this, the application, which executes unwanted system commands, is like a pseudo system shell, and the attacker may use it as an authorized system user.
- Note, the commands are executed with the same privileges as the application and/or web server.
- Command injection attacks are possible in most cases because of lack of correct input data validation, which can be manipulated by the attacker (forms, cookies, HTTP headers etc.

SCREENSHOT:



SQL INJECTION

- SQL injection (also known as SQL fishing) is a technique often used to attack data driven applications.
- This is done by including portions of SQL statements in an entry field in an attempt to get the website to pass a newly formed rogue SQL command to the database (e.g., dump the database contents to the attacker).
- SQL injection is a code injection technique that exploits a security vulnerability in an application's software.
- The vulnerability happens when user input is either incorrectly filtered for string literal
 escape characters embedded in SQL statements or user input is not strongly typed and
 unexpectedly executed.
- SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

INSTRUCTIONS:

- 1. Input the below text into the User ID Textbox (See Picture).
- 2. %' or '0'='0
- 3. Click Submit

NOTES:

- In this scenario, we are saying display all record that are **false** and all records that are **true**.
 - %' Will probably not be equal to anything, and will be false.
 - o '0'='0' Is equal to true, because 0 will always equal 0.
- Database Statement
 - mysql> SELECT first_name, last_name FROM users WHERE user_id = '%' or '0'='0';
- Know the Database name
 - %' or 0=0 union select null, database()
- Display all tables in information_schema
 - o %' and 1=0 union select null, table name from information schema.tables #
- Display all the user tables in information_schema
 - %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
- For more information visit:

http://www.computersecuritystudent.com/SECURITY TOOLS/DVWA/DVWAv107/lesson 6/

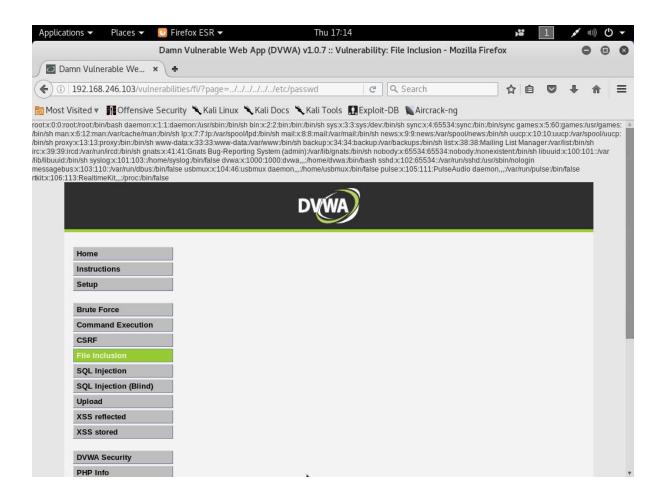
SCREENSHOT:



FILE INCLUSION

- Allows an 'attacker' to include remote/local files into the web application.
- Remote <u>File Inclusion</u> (RFI) and Local File Inclusion (LFI) are vulnerabilities that are
 often found in poorly-written web applications.
- These vulnerabilities occur when a web application allows the user to submit input into files or upload files to the server.
- LFI vulnerabilities allow an attacker to read (and sometimes execute) files on the victim machine.
- This can be very dangerous because if the web server is misconfigured and running with high privileges, the attacker may gain access to sensitive information.
- If the attacker is able to place code on the web server through other means, then they may be able to execute arbitrary commands.
- RFI vulnerabilities are easier to exploit but less common. Instead of accessing a file on the local machine, the attacker is able to execute code hosted on their own machine.

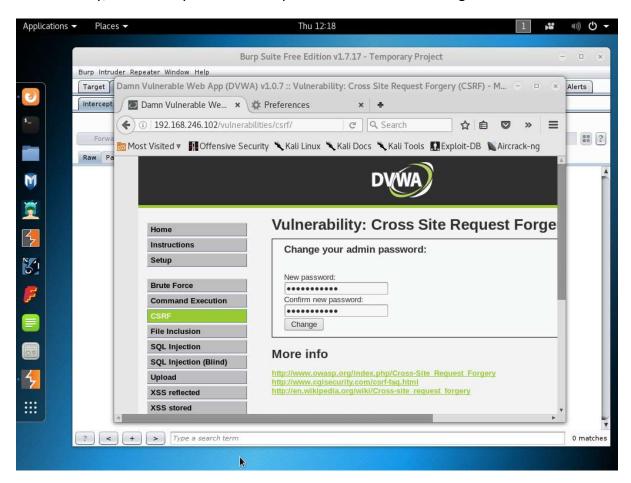
SCREENSHOT:



CSRF/XXS

- Cross-site request forgery, also known as one-click attack or session riding and abbreviated as CSRF (sometimes pronounced sea-surf) or XSRF,
- is a type of malicious <u>exploit</u> of a <u>website</u> where unauthorized commands are transmitted from a user that the web application trusts.
- There are many ways in which a malicious website can transmit such commands;
 specially-crafted image tags, hidden forms, and <u>JavaScript</u> XMLHttpRequests,
- for example, can all work without the user's interaction or even knowledge.
 Unlike <u>cross-site scripting</u> (XSS), which exploits the trust a user has for a particular site,
 CSRF exploits the trust that a site has in a user's browser.

Firstly, User will try to Enter New password and Hits on Change Button.



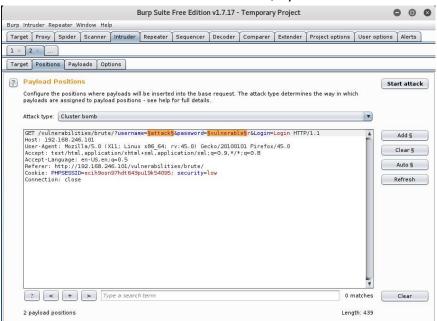
• Secondly, Burp suite will Capture the Request and Alters it as shown in image below: User enters Password as validuser:validuser but we edited as hacker:hacker.



- Query Submitted by Middle-Man will executed Successfully.
- when the Authenticated User tries to Change Password again it throws Error message because data stored in database will be the password entered by Middle-man not by the Valid User.
- Acknowledgement show to User as "Password Changed", shown below

BRUTE FORCE

- A brute force attack is a term in <u>cryptanalysis</u>. It means trying to break
 a <u>coded cyphertext</u> by trying a lot of possibilities with fast <u>computers</u>. For example, a
 large number of the possible <u>keys</u> are tried in the <u>key space</u>. If successful,
 this <u>decrypts</u> the encrypted message.
- The theoretical possibility of a brute force attack is recognized by the <u>cryptographic</u> system designers. They work to make the cryptographic system very difficult for computers to break using brute force attack. For that reason, one of the definitions of "breaking" a cryptographic scheme is to find a method faster than a brute force attack.
- The selection of an appropriate <u>key length</u> depends on how difficult it will be to break it using a brute force attack. By <u>obfuscating</u> the data before <u>encryption</u>, brute force attacks are less effective and more difficult to determine.
- In this challenge we have to find out whether it has been vulnerable to Brute Force Attack and IF yes, then Find out the Username and Password.
- We required a Tool here to Capture the Request-Response transactions of Web Application, We used Burpsuite Here to exploit this Vulnerability.
- Open Burpsuite and Set Proxy to 127.0.0.1:8080
- Open Browser Setting and set proxy to localhost:8080
- Enter any credentials and capture those Packets in Burpsuite under Proxy tab, just forward it to Intruder (note down error reply which has to be entered in Grep-Match field)
- Analyze the Request and find Username and Password text filed enclosed with \$ \$
- Mark username and Password Field with \$ symbol.



- Select Attack Type = Cluster Bomb, Upload Payload 1 and 2
- Under Option tab Set Grep-Match as Incorrect
- Hit Button => Start Attack
- After Matching the Every Payload Entries, Burpsuite gives Username & password (Observes untick on Grep Field)
- Obtained Username and Password is "admin"& "password"

