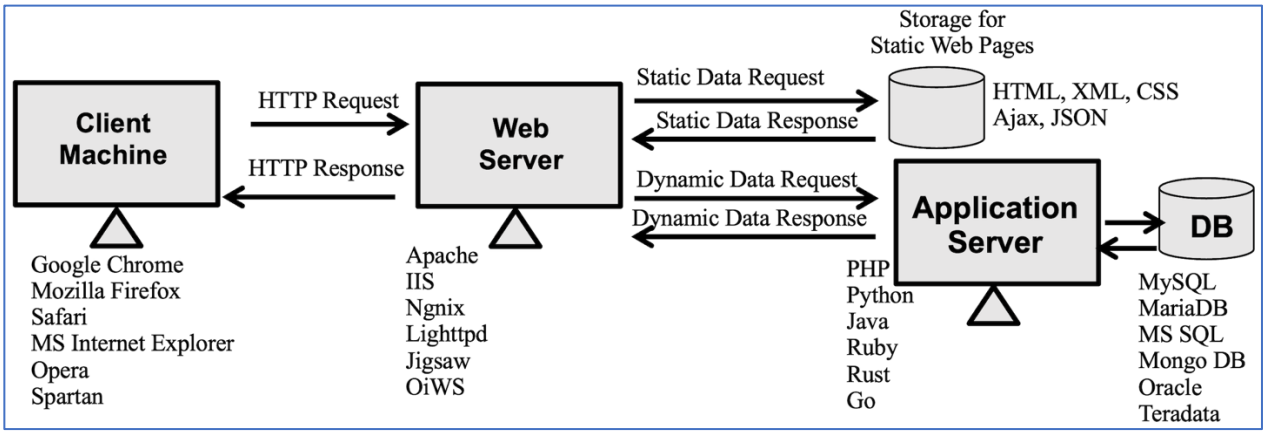
HO#2.9: Web App Penetration Testing - I

Dear students, so far, we have been doing Network Penetration Testing, where we have been identifying and exploiting vulnerabilities in network design, configuration, and protocols. We have used tools like `wireshark`, `nmap`, `nessus`, `Metasploit` and so on to scan IP ranges, network devices, open ports, services, and operating systems.

Today we will switch gear and will start with web application penetration testing, which is the process of identifying, exploiting and assessing vulnerabilities in web applications, logic, input validation, authentication and APIs using tools like `Burp Suite`, `OWASP ZAP`, `Nikto`, `SQLmap` and so on. We will simulate real-world attacks to uncover security flaws such as authentication issues, command injection, SQLi, XSS and so on with the objective to improve the application's security posture and protect sensitive data.

Architecture of a Web Application and Web Application Pen-Testing



Web application penetration testing is the process of evaluating the security of a web applications and web services against malicious cyber-attacks such as Command injection, SQL injection, Cross-Site scripting and so on. It is important because all the web applications are open to the users of the Internet and any one can visit them and check for vulnerabilities and exploit them to gain unauthorized access. You can think of web applications as open doors to your home or business. They include any software application where the user interface or activity occurs online. This can include email, a retail site, or an entertainment streaming service, among countless others. Since websites must allow traffic to come and in and out of the network, hackers often attack the most commonly used ports. This includes:

- Port 80 (HTTP): For unsecured website traffic
- Port 443 (HTTPS): For secured website traffic
- Port 21 (FTP): The file transfer protocol for transferring files to and from your servers
- Ports 25 (SMTP), and Port 110 (POP3): Email protocols often used by organizations to send and receive email.

The website hosted on a web server is accessible to anyone who knows its IP or URL. The problem is the attacker is not just interested in the website for its simple usage for which that application is intended rather the attacker will try to execute different types of attacks.

OWASP Top 10 Vulnerabilities

Web application vulnerabilities involve a system flaw or weakness in a web-based application, largely due to *not validating/sanitizing form inputs, misconfigured web servers, and application design flaws.* **OWASP** (Open Web Application Security Project) is a non-profit organization focused on improving the security of software. In the context of cybersecurity, OWASP is widely known for providing free resources, tools, and guidelines to help developers, businesses, and security professionals create secure web applications. The **OWASP Top 10** is a list of the most critical security risks for web applications. Here's a simple breakdown of each vulnerability.

2010	2013	2017	2021
A-1 Injection	A1-Injection	A1-Injection	A1-Broken Access Control
A2- Cross-Site Scripting	A2- Broken Authentication	A2- Broken Authentication	A2- Cryptographic Failure
A3- Broken Authentication	A3- Cross-Site Scripting	A3- Sensitive Data Exposure	A3- Injection
A4- Insecure Direct Object References	A4- Insecure Direct Object References	A4- XML External Entities	A4- Insecure Design
A5- Cross-Site Request Forgery	A5- Security Misconfiguration	A5- Broken Access Control	A5- Security Misconfiguration
A6-Security Misconfiguration	A6- Sensitive Data Exposure	A6- Security Misconfiguration	A6- Vulnerable and outdated components
A7- Cryptographic Failures	A7- Missing Function Level Access Control	A7- Cross-Site Scripting	A7- Identification and Authentication failures
A8- Failure to restrict URL access	A8- Cross-Site Request Forgery	A8- Insecure Deserialization	A8- Software and Data Integrity Failures
A9- Insufficient Transport Layer Protection	A9- Using Components with known vulnerabilities	A9- Using. Components with known vulnerabilities	A9- Security Logging and Monitoring Failures
A10- Unvalidated redirects and Forwards	A10- Unvalidated Redirects and Forwards	A10- Insufficient Logging and Monitoring	A10- Serer-Side Request Forgery

1. **Broken Access Control:** When users can do things they're not supposed to, like accessing other users' data or performing admin actions without permission.
2. **Cryptographic Failures:** Sensitive data isn't protected properly, like passwords or credit card numbers being sent or stored without encryption, making them easy to steal.
3. **Injection:** Malicious data is sent to a program (like shell commands, code, or SQL) to trick it into doing something harmful, such as leaking sensitive data or taking control.
4. **Insecure Design:** The application is built in a way that doesn't consider security from the start, making it vulnerable to attacks.
5. **Security Misconfiguration:** When systems or applications are set up incorrectly, leaving gaps for attackers to exploit. This includes not keeping software updated or using default passwords.
6. **Vulnerable and Outdated Components:** Using old or vulnerable libraries or software in your application, which attackers can use to break in or exploit weaknesses.
7. **Identification and Authentication Failures:** Weak or broken methods for checking who a user is, allowing attackers to pretend to be legitimate users and access data they shouldn't.
8. **Software and Data Integrity Failures:** Failing to ensure that software and data haven't been tampered with, like installing a compromised update or allowing untrusted data to influence system behaviour.
9. **Security Logging and Monitoring Failures:** The application or system isn't keeping track of important security events (like login attempts) or responding to attacks in real-time, allowing attackers to go unnoticed.
10. **Server-Side Request Forgery (SSRF):** The application can be tricked into making requests to other servers, allowing attackers to access sensitive information or interact with systems that should be off-limits.

Key Objectives of Web Application Penetration Testing

- **Identify Vulnerabilities**: Detect flaws and weaknesses in the web application that could be exploited.
- **Assess Impact**: Determine the potential impact of exploiting these vulnerabilities.
- **Verify Exploits**: Confirm the existence of vulnerabilities by safely exploiting them.
- **Report Findings**: Provide detailed reports on vulnerabilities and recommend mitigation strategies.
- **Improve Security Posture**: Help developers and administrators improve the security of web applications.

Gathering Information about Web Applications

We have covered the phases of Info Gathering, Scanning and Vulnerability Analysis in our initial handouts. You must not forget to perform these steps, while doing web application penetration testing as well. The objectives of these three steps are:

- **Identify Entry Points:** Uncover potential points of attack such as login forms, admin panels, and public directories.
- **Understand Technologies:** Determine the technologies and platforms used (e.g., web server, CMS, frameworks), which helps in identifying specific vulnerabilities.
- **Map the Site Structure:** Create a map of the website's structure, which aids in navigating and targeting specific areas during a penetration test.
- **Discover Subdomains and Services:** Find additional subdomains and services that might be overlooked but are part of the target's attack surface.
- **Gather Intelligence:** Collect information about the website's owners, IP addresses, DNS records, and other metadata that can be useful for social engineering attacks or more targeted exploits.

Note: There exist several tools that can be used in the different phases of web app pen-testing. A sample workflow along with tools can be:

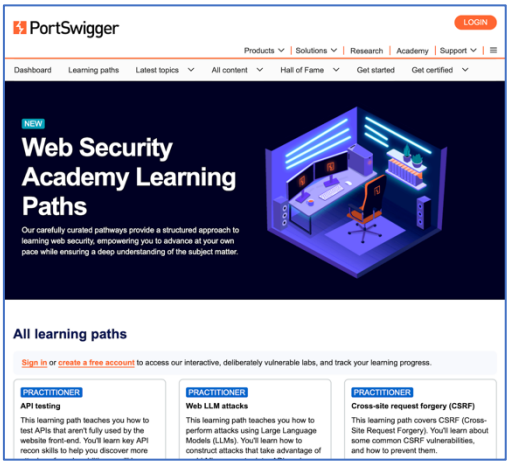
- Reconnaissance: `Google Dorking`, `showdan`, `theHarvester`, `whatweb` and so on.
- Scanning and Vulnerability Analysis: `burpsuite`, `dirb`, `nikto`, and so on.
- Exploitation: `burpsuite`, `hydra`, `SQLMap`, `MSF`, `BeEF` and so on.
- Post Exploitation: `burpsuite`, `MSF`, `BeEF` and so on.

Freely Available Insecure Web Applications

Dear students, in order to practice web application pen testing, we need an actual website to run our tools on. We cannot run them on any live website as it is illegal, however, there exist different options to practice web application pen testing in a controlled environment.

- **Option 1:** Use one of the following freely available deliberately insecure web applications:
  - **Damn Vulnerable Web Application:** https://www.dvwa.co.uk/
    - Frontend: HTML, CSS, JavaScript
    - Backend: PHP
    - Database: MySQL
    - Webserver: Apache
  - **WebGoat:** https://owasp.org/www-project-webgoat/
    - Frontend: HTML, CSS, JavaScript
    - Backend: Java (Spring Framework)
    - Database: Uses Hyper SQL Database (HSQLDB)
    - Webserver: Apache Tomcat or any other Java Servlet Container
  - **Juice Shop:** https://owasp.org/www-project-juice-shop/
    - Frontend: Angular (A modern JavaScript framework)
    - Backend: Node.js (Express Framework)
    - Database: SQLite by default, but support other databases like MongoDB
    - Webserver: Built-in Node.js server
    - Online instance is available at https://juice-shop.herokuapp.com

- **Option 2:** https://portswigger.net/web-security/learning-paths
  Create an account on *PortSwigger Web Security Academy,* which is a free online training platform created by PortSwigger, the developers of Burp Suite. It is designed to help individuals learn web application security concepts, vulnerabilities, and exploitation techniques. There are tons of hands-on exercises in the form of online labs where users can practice exploiting vulnerabilities in a controlled, legal and realistic environment.



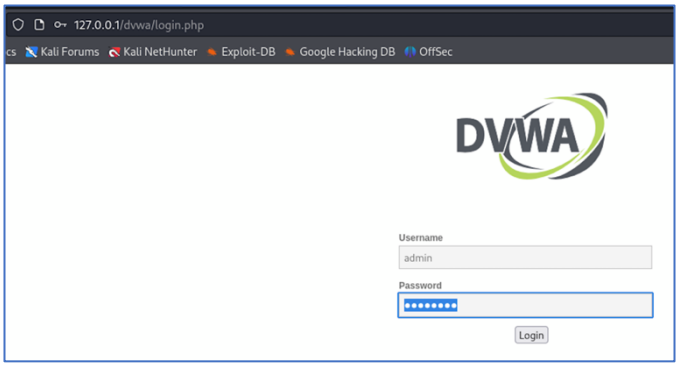
- **Option 3:** https://tryhackme.com
  Create an account on *TryHackMe*, which is an online platform designed for learning cybersecurity through hands-on practice in an interactive gamified environment. Users can enhance their knowledge and skills in various areas of cybersecurity, like web application security, network security, cloud security, cryptography, malware analysis, and reverse engineering. It provides CTF-style challenges where users solve puzzles, exploit vulnerabilities, and submit "flags" as proof of success.

Damn Vulnerable Web Application (DVWA)

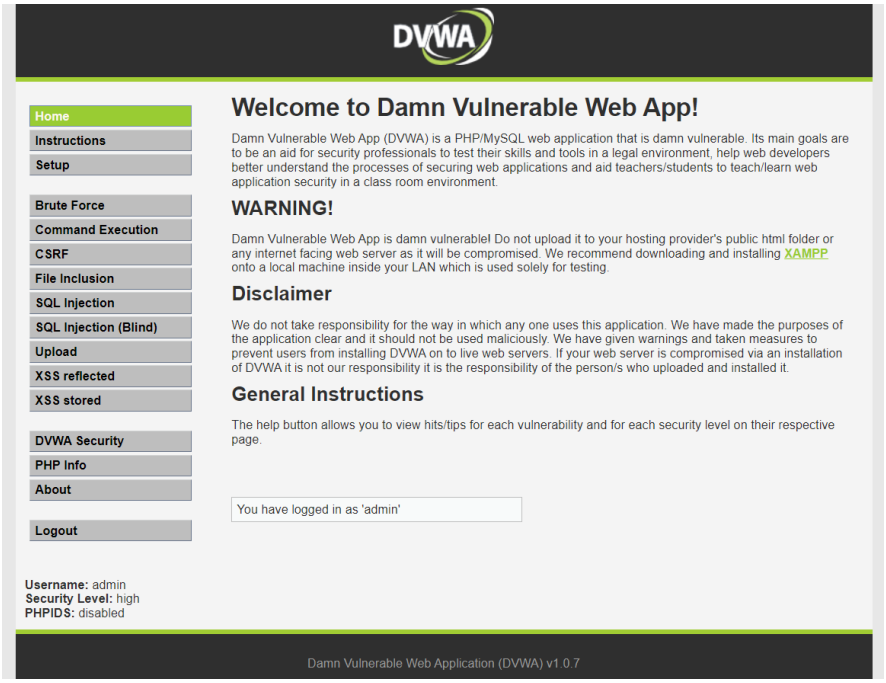
DVWA is a PHP/MySQL web application intentionally designed to be vulnerable. It provides a way of teaching application developers about the common programming mistakes that allow malicious code to be inserted into strings, making the application unsafe for users. You can think of it as a playground for the most common web vulnerabilities, ranging from easy to hard, all within a simple, straightforward interface.

Lab Environment Setup (On Metasploitable2)

- The Metasploitable2 machine has an already installed a Damn Vulnerable Web Application.
- So, with Metasploitable2 and Kali Linux machine running inside your Virtual Box, open a browser inside Kali and type the IP of Metasploitable2, which will open this website via http over port 80 as shown:
- Click DVWA link and it will open a php login page.



- After giving the username as *admin* as password as *password*, it will take you to Damn Vulnerable Web App having lot of vulnerable applications like BruteForce, Command Injection, XSS, SQLi, CSRF etc, where you can practice all these attacks.



Lab Environment Setup (On Kali)

Remember, while running this program your machine will be extremely vulnerable to attack. You should disconnect from the Internet while using this program. So better to install the DVWA application on your Kali machine and set the configuration as to bind it to localhost to minimize the exposure. The DVWA download is available free of charge to everyone under the terms of the GNU General Public License. It's perfect as a self-contained long-term penetration testing lab, and it's open-source so that you can add, subtract, and customize its vulnerabilities. Follow following steps in order to install DVWA application inside your Kali Linux machine

Installing Apache and PHP: We have installed and used Apache Web Server and PHP in our Handout#1.3. For details refer to that handout, however, the commands to install are given below:

```
# apt-get install apache2 apache2-doc
# apt-get install php libapache2-mod-php php-mysql php-mcrypt
# apache2 -v
Server version: Apache/2.4.62 (Debian)
# php -v
PHP 8.2.21 (cli) (built: Jul 25 2024)
# systemctl start apache2.service
# netstat -ant | grep 80
tcp6       0      0 :::80                   :::*                    LISTEN
```

Installing MySQL (mariadb-server): Both MySQL (owned by Oracle) and MariaDB (community driven fork of MySQL) are open-source RDBMS, but they differ in terms of licensing, features, and performance. Both stores and manages data using SQL (Structured Query Language) and supports multi-user access to databases.

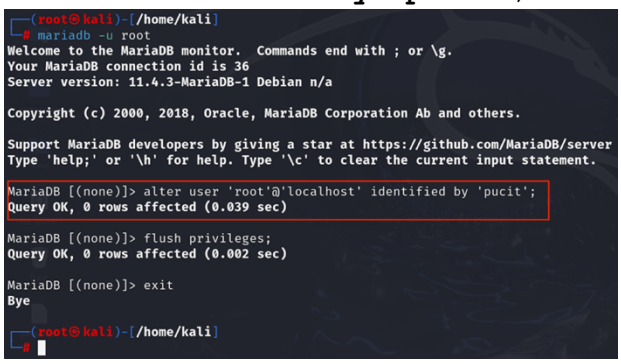
```
# apt-get install mariadb-server
# mysql --version
mysql from 11.4.3-MariaDB, client 15.2 for Debian-linux-gnu
# mariadb --version
mariadb from 11.4.3-MariaDB, client 15.2 for Debian-linux-gnu
# systemctl start mariadb.service
# netstat -ant | grep 3306
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
```

The configuration file of Mariadb is `/etc/mysql/mariadb.conf.d/50-server.cnf`. By default, MySQL only listens on local host for security reasons, as shown in the above output. If you want to allow remote connections, open its configuration file with root privileges, and comment the line `bind-address = 127.0.0.1` by putting a hash symbol before it. After this. You need to restart the service.

```
# systemctl restart mariadb.service
# netstat -ant | grep 3306
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp6       0      0 :::3306                 :::*                    LISTEN
```

The root user is the default super user created during MySQL or Mariadb installation. For the first time when you login as root you use the following command to set the password of root user:

```
      # mariadb -u root
MariaDB [(none)]> alter user 'root'@'127.0.0.1' identified by 'pucit';
Query OK, 0 rows affected (0.039 sec)
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.002 sec)
MariaDB [(none)]> exit
Bye
```



Installing DVWA: On Kali Linux as root user follow the following steps to download the DVWA repository from `github` directly inside the `/var/www/html/` directory using following commands:

```
# cd /var/www/html/
# git clone https://github.com/digininja/DVWA.git
# mv DVWA dvwa
# chmod -R 777 dvwa
```

After downloading, the next step is to configure this web application. For this we need to make minor modification in its configuration file `/var/www/html/dvwa/config/config.inc.php.dist`. First make a copy of this file with .php extension, then open it in an editor. Let us change the db_user to 'arif' and db_password to '123456' in this configuration file.

```
# cp dvwa/config/config.inc.php.dist config.inc.php
# vim config.inc.php
```

```
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
#   See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ]   =  '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ]     = 'arif';
$_DVWA[ 'db_password' ] = '123456';
$_DVWA[ 'db_port']      = '3306';

# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module
#   You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ]  = '';
$_DVWA[ 'recaptcha_private_key' ] = '';

# Default security level
```

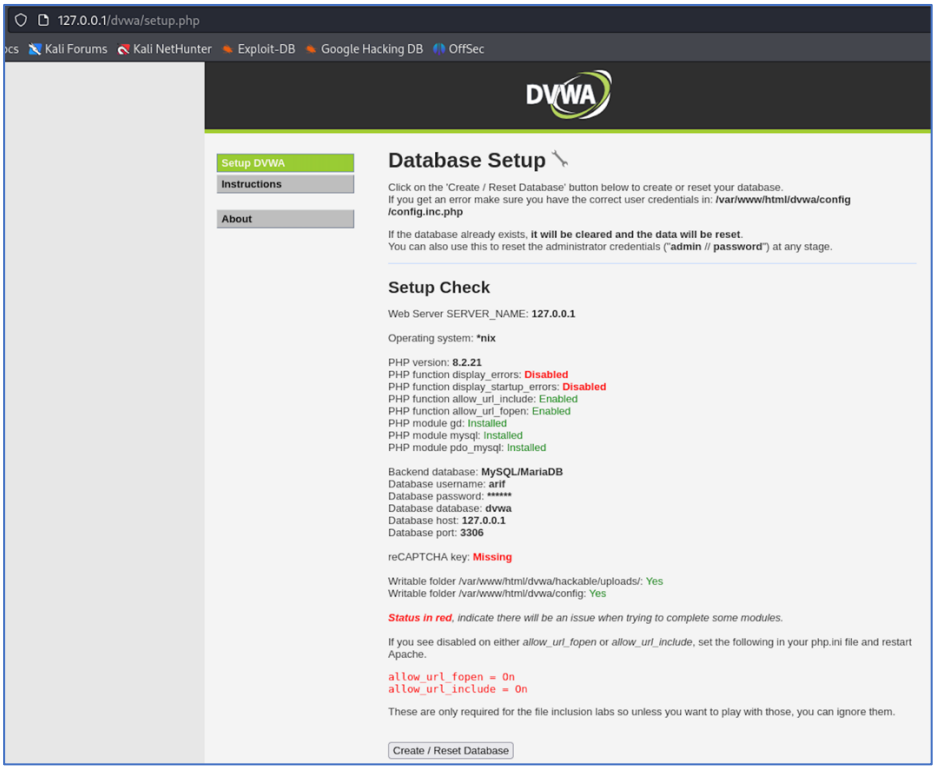
Let us restart the Mariadb, and create a user `'arif'` with the password is `'123456'`. Remember that this username and password should exactly be the same as the password and username we have entered in the configuration file (`config.inc.php`) of DVWA web application as shown above.

```
# systemctl start mariadb.service
# mariadb -u root -p
MariaDB [(none)]>create user 'arif'@'%' identified by '123456';
MariaDB [(none)]>grant all privileges on dvwa.* to 'arif'@'%' with grant option;
```

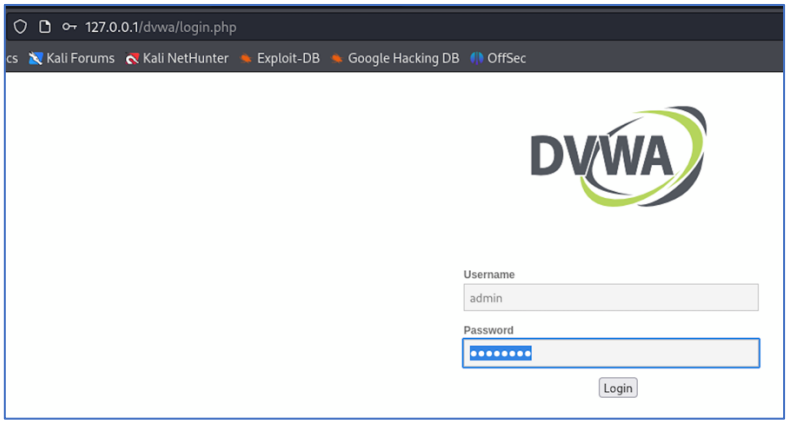
Finally, we have finished the work of database, now we need to configure the `php.ini` file located in the `/etc/php/8.2/apache2/` directory. Open this file in an editor and set the two values of *allow_url_fopen* and *allow_url_include* to **On**. Save and close the file and restart Apache2 service:

```
# systemctl restart apache2.service
```

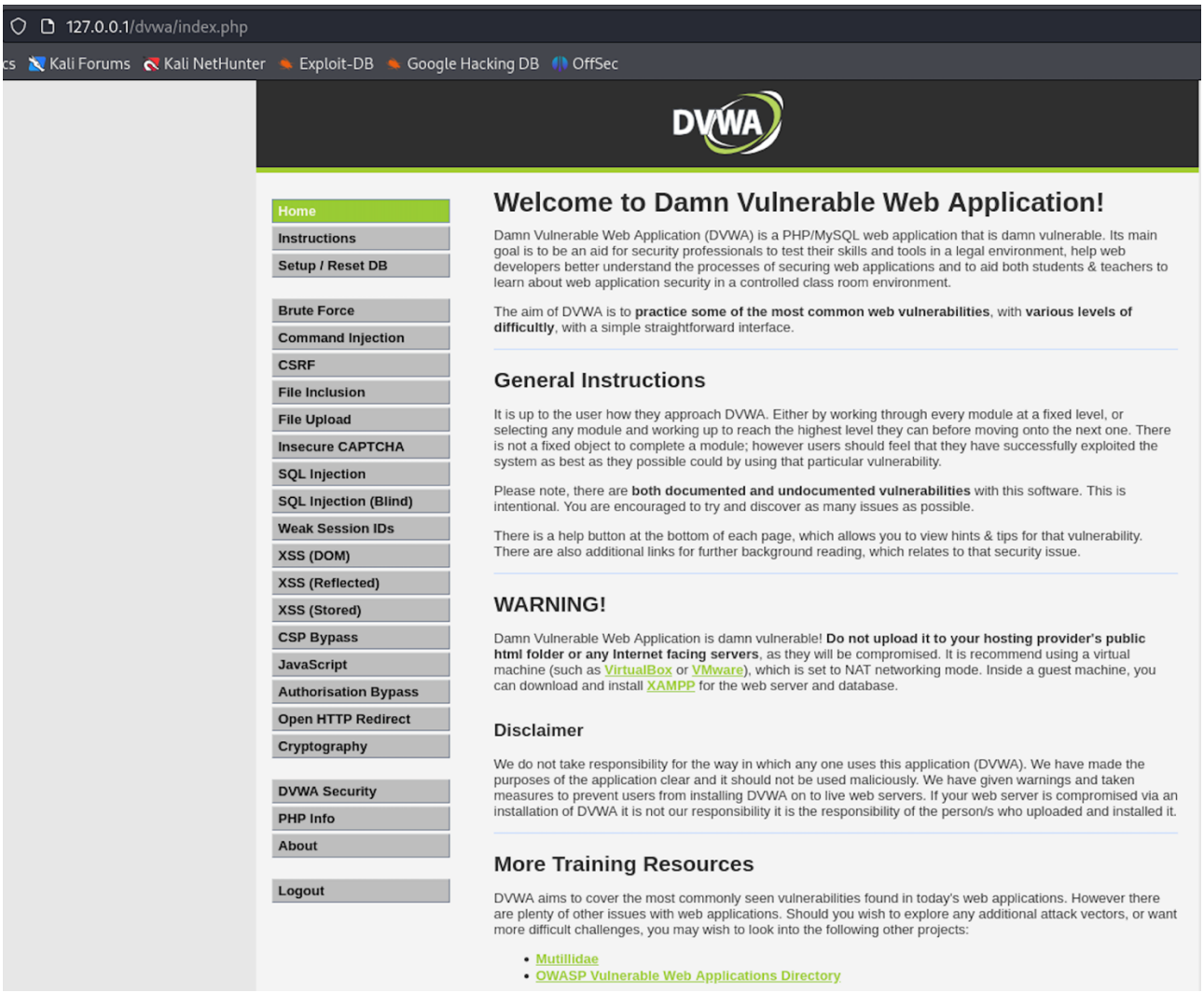
Let's open the browser and navigate to `http://127.0.0.1/dvwa/setup.php` , which will open the following page.



In the above page, just scroll down and click on "Create/Reset Database", which will create and configure the database and we will be redirected to DVWA login page, where you can login by giving the credentials `admin:passowrd` as shown below:



After you have successfully logged in, you will get the main page of DVWA as shown below:



In the left pane, you there are different links to vulnerable pages corresponding to different attacks. Moreover, DVWA offers four options to set your desired pentesting difficulty level:

- **Low**: This security level is completely vulnerable and has no security measures at all.
- **Medium**: This setting mainly demonstrates bad security practices by showing how the developer has tried but failed to secure an application.
- **High**: This option is an extension of the medium level. The vulnerabilities at this level put an upper limit on your exploitation, similar to how you'd handle various Capture-The-Flags (CTFs) competitions.
- **Impossible**: This is the default setting for the Kali Linux installation. Although the developers claim that the Impossible level should be secure against all vulnerabilities, but to me absolute cyber security doesn't exist due to undocumented vulnerabilities and zero-day attacks.

- Before you start, please click the **Instructions** button in the left pane, and give a bird's eye view to the page contents.
- Moreover, while practicing different types of attacks, do click the **View Help** and **View Source** button in the right bottom of every attack page, to have a crystal-clear understanding as to how the attack works. ☺

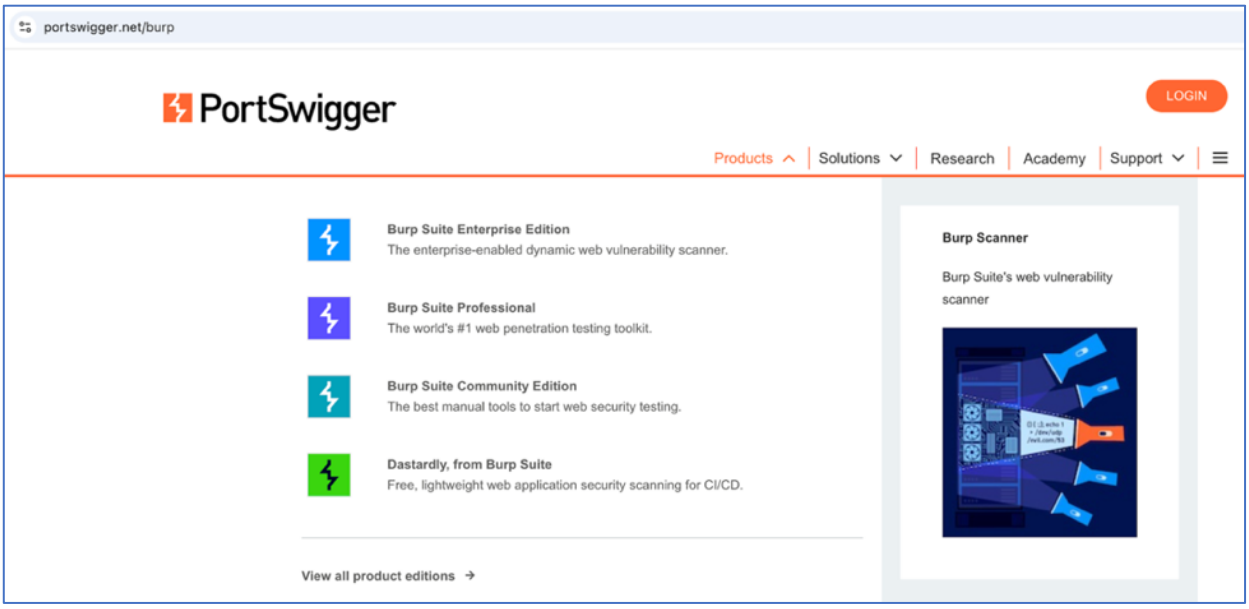
Types of Attacks that can be Performed on DVWA

- **Injection attacks** are a class of vulnerabilities where untrusted data is inserted or "injected" into a web application, altering the execution flow or behavior of the application. There exist numerous types of injection attacks, however, the five main types are:
    - **Command Injection:** The attacker executes arbitrary OS commands on the host OS via a vulnerable application. It can lead to OS-level control, allowing attackers to execute arbitrary commands, access system files, or compromise the server. For example, injecting the command `rm -rf /` into a form that allows input to be passed to the system shell.
    - **Code Injection:** Malicious code is injected into an application, which the application then executes as part of its normal flow. Will be covered in next Module.
    - **SQL Injection (SQLi):** An attacker manipulates SQL queries by injecting malicious SQL code via input fields to access or modify the database. It allows unauthorized access to database contents, data modification/deletion, or may be complete access to database server.
    - **File Inclusion:** Local File Inclusion (LFI) and Remote File Inclusion (RFI) attacks, which allow an attacker to include files from the local server or remote servers, potentially leading to arbitrary code execution.
    - **Cross-Site Scripting (XSS):** Malicious scripts (usually JavaScript) are injected into web pages, which get executed in the browser of other users who view the page. *It can lead to session hijacking, data theft, or redirection to malicious websites.*
        - **Stored XSS**: Inject scripts into the application that get stored on the server and later gets executed whenever another user views the infected page. For this the user need not to click any link.
        - **Reflected XSS**: Inject scripts via URLs or forms that are not stored on the server. Java script code doesn't get executed by just visiting the infected web page, rather will work only when the user clicks a link pointing to malicious web page and the code gets executed on the user machine.
        - **DOM-based XSS**: Exploit client-side JavaScript code execution vulnerabilities.
- **Authentication and session management attacks** target flaws in authentication, session handling, and user management. DVWA offer following types of such attacks:
    - **Brute Force:** In brute force attack the attacker send a lot of usernames and lot of passwords and hope that by accident he/she might hit the correct one. You usually perform this attack to see if the target machine has default credentials or weak passwords
    - **Weak Session IDs:** Exploit weak session management practices to hijack user sessions by predicting or capturing session identifiers.
    - **User Enumeration:** Identify valid usernames by analyzing different server responses for valid and invalid users.
- **Security Misconfiguration attacks**
    - **Insecure CAPTCHA**: Bypass CAPTCHA mechanisms, often used to prevent automated attacks, and prove their weaknesses.
    - **Security Misconfiguration**: Leverage poor configuration practices like displaying error messages, exposing server version details, or using default credentials.

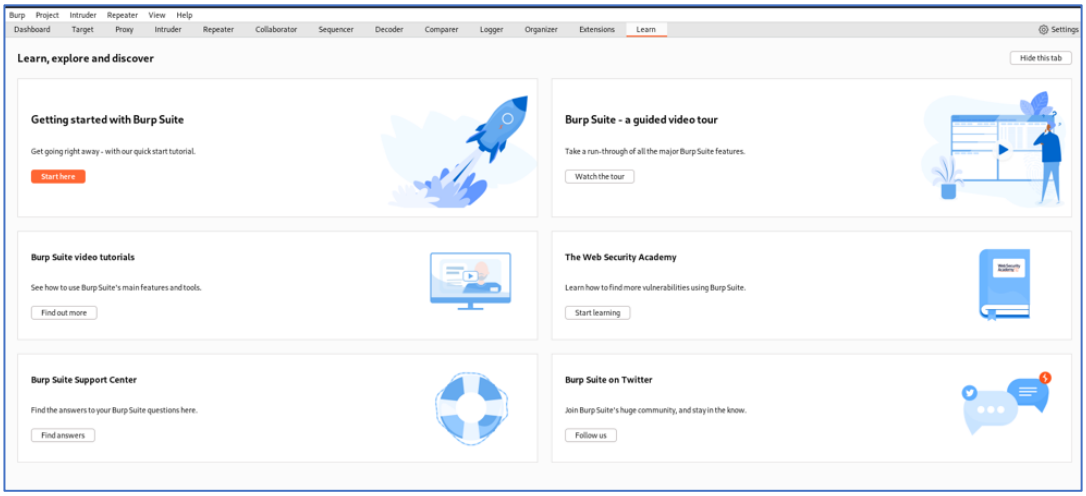
- **CSRF (Cross-Site Request Forgery)**: Trick a user into executing unwanted actions on a web application where they're authenticated, often leading to actions like changing passwords or making requests on their behalf. An example of such attack is the attacker sends a user a link that when clicked, changes the user's email address on a banking site without their consent.
- **File Upload**: Exploit insecure file upload functionality to upload malicious files, such as PHP shells, enabling further attacks like remote code execution.
- **Password Hashing**: Exploit weak password hashing algorithms (e.g., MD5) by cracking hashed passwords.

Burp Suite

Dear students, when we started with Internetworking with Linux in our initial Handout#1.3, we started with a tool called **`wireshark`**, which is a network protocol analyzer that captures and analyzes network traffic (TCP, UDP, HTTP etc.) in real-time. The most commonly used tool used for web applications penetration testing is **`Burp Suite`**, which is a web application penetration testing framework developed by PortSwigger (https://portswigger.net/burp). It is a java executable, cross-platform, and has become an industry standard suite of tools used by information security professionals to *identify vulnerabilities and verify attack vectors for web-based applications*. Burp Suite comes with an Enterprise, Professional and Community Edition. Fortunately, Kali Linux comes with pre-installed free community edition of Burp Suite ☺.

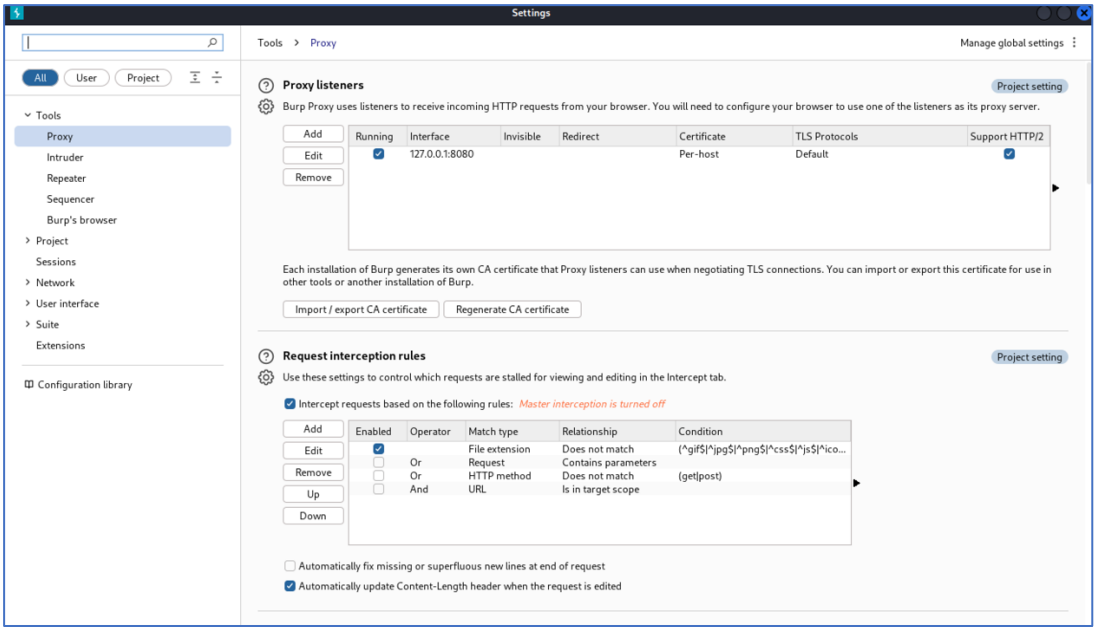


Using Burp, we can *analyze* HTTP requests, can *modify* them and *resend* the modified requests. Being able to edit requests that our browser sends is the key to testing a web application. So, this is the core thing (Intercept) that Burp do, other than many other fancy tasks like Repeater (repeats requests), Intruder (brute force tools), Fuzzer and so on.
Following screenshot is the main screen of Burp Suite, where you can see different tabs *Proxy*, *Target*, *Repeater*, *Intruder*, *Decoder*, *Comparer*, *Dashboard* and so on depending on whether you have Enterprise, Professional or Community Edition of Burp Suite.

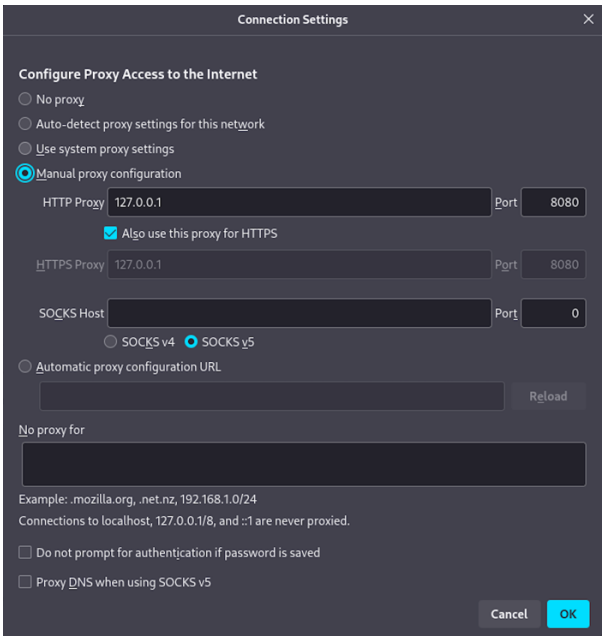


Configuring Burp Suite:

- **Step 1:** The first thing you need is to run a proxy listener inside Burp to accept a request from your browser (Firefox). To do this inside Burp, **click the Settings** gear icon in the top right, click Proxy in the left pane, which will open a new window as shown below. Check out the Proxy listeners, and add a listener at 127.0.0.1:8080, as shown in following screenshot:



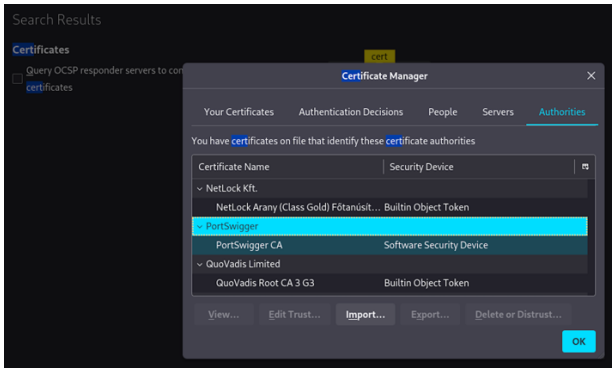
- **Step 2:** The second thing you need to do is to configure your browser (Firefox) to use the proxy listener as its proxy server. To do this, open the **Settings of Firefox** by clicking the "open application menu" icon in the top right corner of your browser window. Click Settings, select General and scroll to the bottom of the page to Network Settings and click Settings button. This will open the Connection Settings dialogue as shown below. Instead of No proxy radio button, click the Manual proxy configuration and set HTTP Proxy to 127.0.0.1 with Port 8080 and click OK button. An alternative is to use Foxy-Proxy (browser extension) to change the proxy settings with a single click. If you want Burp to stop capturing traffic, you can turn off the proxy settings inside your browser.



Using Burp Suite and Firefox:

- **Capturing HTTP Traffic inside Burp:**
  - If you have performed all the above steps correctly, your Burp would have been intercepting all the outgoing and incoming web traffic via your Firefox browser.
  - Try intercepting HTTP traffic inside Burp by visiting some http web site say our favorite http://scanme.nmap.org/. This will work, to verify just go on the Target tab of Burp, where you can see the incoming and outgoing HTTP request/response messages ☺

- **Capturing HTTPS Traffic inside Burp:**
  - Let us now repeat the same for an https website say our favorite https://arifbutt.me. This time, your browser (Firefox) may not be able to open the web page and probably will give you a message saying that your connection is not secure. This is because our Firefox do not trust Burp Suite ☹. So, in order to intercept the HTTPS traffic inside Burp, we need to install/import the Port Swigger's Certificate on your browser (Firefox).
  - **What is CA certificate:** A CA (Certificate Authority) certificate is a digital certificate issued by a trusted organization like GoDaddy, DigiCert, and GlobalSign to name a few. The primary role of the CA is to verify the identity of entities (such as websites, organizations, or individuals) and then issue digital certificates that attest to that identity.
  - **Delete Previously installed *PortSwigger* Certificate (Optional):** To do this, open the Settings of Firefox by clicking the "open application menu" icon in the top right corner of your browser window. In the search text box search for *certificates*, Click View Certificates button, and this will open up the Certificate Manager Window. Look if there exist a *PortSwigger* certificate and delete if it is there.
  - **Download *PortSwigger* Certificate:** Inside the Firefox type http://burp, which will open an almost blank Burp Suite Community Edition web page. In the top right corner of this page, click the link "CA Certificate", which will download a certificate file named `cacert.der` inside the Download directory.
  - **Import CA certificate in Firefox:** In Firefox, open the Settings of Firefox by clicking the "open application menu" icon in the top right corner of your browser window. In the search text box search for *certificates*, Click View Certificates button, and this will open up the Certificate Manager Window. Click Import button, browse to the download directory and select `cacert.der` file and click OK after selecting the check box saying "Trust this CA to identify websites" and "Trust this CA to identify email users".

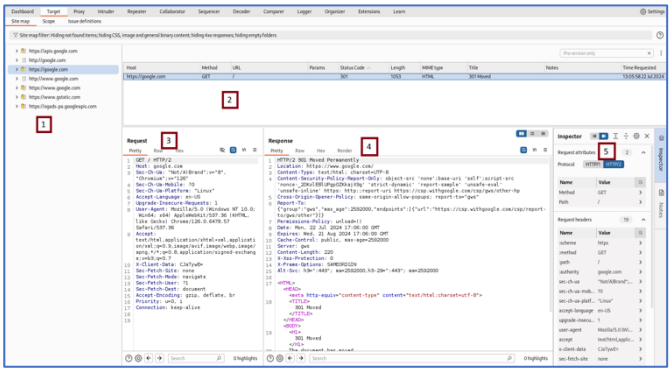


  - Now try intercepting HTTPS traffic inside Burp by visiting some https web site say our favorite https://arifbutt.me. This will work now, to verify just go on the Target tab of Burp, where you can see the incoming and outgoing HTTP request/response messages ☺

Target Tab of Burp Suite: *It provides an organized view of all captured traffic and serves as a central workspace for managing and analyzing the application's attack surface. The Target tab captures traffic even when the intercept is turned off in the Proxy tab and is a READ ONLY interface.*

- **Site map sub tab** displays a hierarchical tree of the target website or application discovered by Burp during scanning and browsing, having five panes:

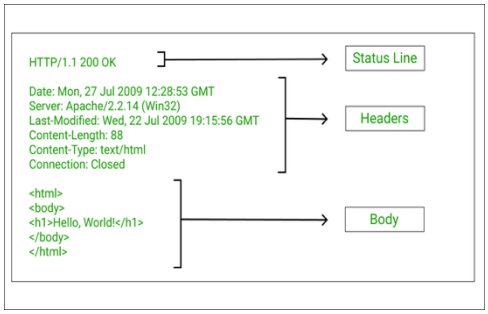
1. On the left the **Site map** pane displays a hierarchical tree of the target application, showing directories (folder icons), files (document icon), parameterized requests (gear icon) and individual URLs that make up the site. As you browse or scan, Burp Suite dynamically update the site map with all the discovered resources.



2. The **Contents section** show hosts, method, URL, params, status code and length of all the HTTP requests that we performed with the selected website in the left pane. You may find some links in black color (properly visited) or gray color (not visited but opened).

3. The **Request section** displays detailed information about the HTTP request object, currently selected in the Contents section. The structure of HTTP request is shown:
   - *Status Line:* Specifies the request method (GET, POST), the requested resource URL and protocol version.
   - *Headers:* Provide metadata about the request as key:value pairs, e.g., host, user-agent, content-type, cookies and so on.
   - *Body:* Contains optional data being sent by the client to server, such as form data or file uploads (for POST method).
   - *Query Parameters:* Passed in the URL after a **?** symbol, as key:value pairs separated by **&** symbol.

4. The **Response section** displays detailed information about the HTTP response object, currently selected in the Contents section. The structure of HTTP response is shown:
   - *Status Line:* Includes the HTTP version, a status code of outcome of request like (1xx: Informational, 2xx: Success, 3xx: Redirection, 4xx: Client Errors, 5xx: Server Errors), and a textual description of the status code.
   - *Headers:* Provide metadata about the response as key:value pairs, such as date, server type, content type, content length, set etc.
   - *Body:* Contains optional response data sent by the server, e.g., an HTML, JSON, an image, or may be an error message.



5. The **Inspector** pane allows detailed examination and modification of HTTP requests and responses in a structured format.

- **Scope sub-tab** helps you define the scope of your test by including or excluding certain URLs, domains or parts of the application that you want to include or exclude from your testing. This helps you focus on relevant areas and avoid testing unintended sections of a web application. To do this either right click the URL in the left pane of sitemap and click add to scope. Or go to the scope subtab, and add the URL(s) you want to add in the scope of your testing.
- **Issue definitions sub-tab** list known vulnerabilities and security issues with explanations and guidance for fixing them.

To Do:

Keep the intercept OFF under Proxy tab, visit different URLs, capture the traffic, and understand how the Target tab works:

- Visit your favourite website **https:arifbutt.me,** visit different pages and see how the sitemap of target tab gets populated and how get and response object are exchanged between the browser and the web server.

- Visit our basic website **http://<IP of M2>/basicloginapp1/index.html** deployed on Metasploitable2, and send the credentials to login (`arif:kakamanna`). It is using the GET method to send the form parameters (inside the URL). The limitation of using the HTTP GET method are sends data as query parameters appended to the URL, is less secure and limited to typically 2048 characters in modern browsers.

```
//index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=de
    <title>Login</title>
</head>
<body>
    <h2>Login Form</h2>
    <form action="login.php" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

Login Form

Username:

Password:

Login

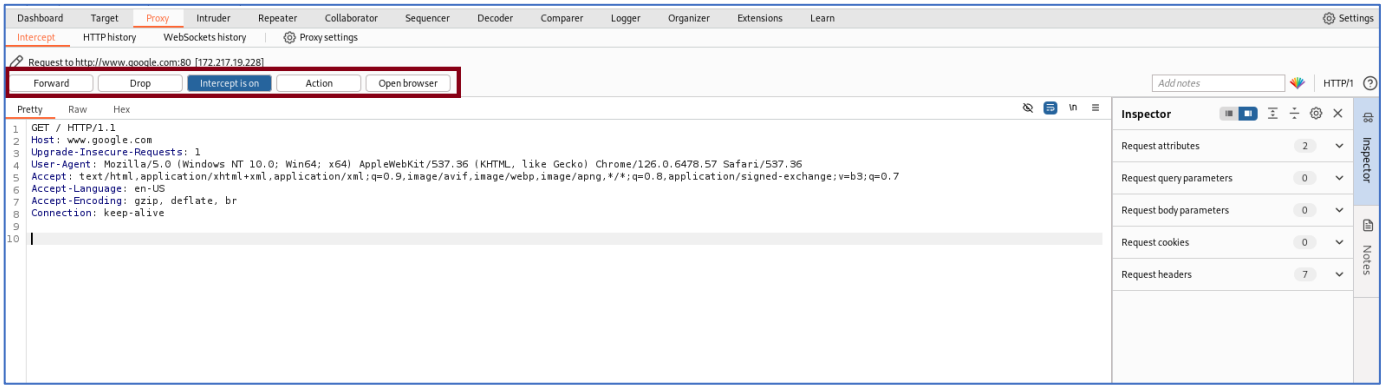
```
//login.php
<?php
// Hardcoded username and password for simplicity
$valid_username = "arif";
$valid_password = "kakamanna";
// Retrieve input from the form
$username = $_POST['username'];
$password = $_POST['password'];
// Authentication
if ($username === $valid_username && $password === $valid_password)
    echo "Login successful";
else
    echo "Invalid username or password.";
?>
```

- Visit our basic website **http://<IP of M2>/basicloginapp2/index.html** deployed on Metasploitable2, and send the credentials to login (`arif:kakamanna`). It is using the POST method to send the form parameters. Note, the form data is sent to the server inside the body of HTTP request, however, data is not encrypted unless HTTPS is used.

- Visit the DVWA website **http://<IP of M2>/dvwa/login.php** deployed on Metasploitable2 machine, and send the credentials to login (admin:password). It is using the POST method to send the form parameters. Do note that on giving correct credentials, you move from **login.php** to **index.php** page of the website.

Proxy Tab of Burp Suite: The Proxy tab with **Intercept turned ON** allows you to pause and manually inspect or modify each request before forwarding it to the server. If Intercept is OFF, Burp still captures and logs all traffic passing through its proxy server, and this data appears in the HTTP history within the Proxy tab and is also reflected in the Target tab's Site Map. So, the Proxy tab is used for real-time interception and modification of requests or responses as they are sent or received by the application.

- The **Intercept** sub-tab, lets you intercept and modify HTTP/S requests and responses. You can pause traffic, examine, and manually edit data.
  - The *Intercept is on/off* button is used to intercept the traffic.
  - The *Forward* button is used if the Intercept is ON, to send the intercepted HTTP/S request from Burp Suite to the target server.
  - The *Drop* button is used if the Intercept is ON, to discard an intercepted HTTP/S request or response.
  - The *Action* button provides additional options and actions that can be performed on the intercepted request or response, like send to intruder/repeater/sequencer.
  - The *Open browser* button is to open the default configured browser.



- The **HTTP History** sub-tab shows a log of all HTTP requests and responses that have passed through the proxy. Useful for reviewing all interactions with the web application.
- The **WebSockets History** sub-tab logs WebSocket messages, allowing you to analyze real-time, full-duplex communications used in modern applications.
- The **Proxy settings** sub-tab opens a new window to configure settings related to the proxy.

To Do:

Make intercept ON under Proxy tab, perform an action in the browser, the request will be intercepted, modify the parameters, headers or body directly in the Proxy tab, and then forward the modified request to the server using the Forward button. Practice these on following web sites:

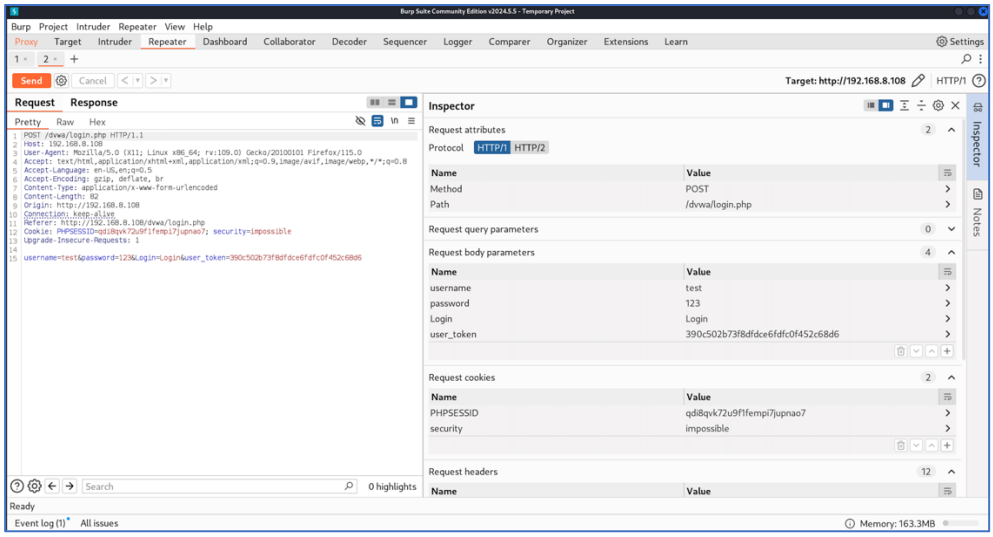
- **http://<IP of M2>/basicloginapp1/index.html**
- **http://<IP of M2>/basicloginapp2/index.html**
- **http://<IP of M2>/dvwa/login.php**

Repeater Tab of Burp Suite: The *Repeater* tab is designed for manual, iterative testing of individual requests. It allows you to modify and resend requests repeatedly to test different scenarios. Later you observe the server's response to identify vulnerabilities. It is useful for:

- Testing input validation (e.g., tampering with parameters).
- Experimenting with authentication or session management.
- Identifying vulnerabilities such as SQL injection, XSS, etc.

Step 1: Send a Request to Repeater

- Make intercept ON under Proxy tab, visit website **http://<IP of M2>/dvwa/login.php** deployed on M2 machine. Give wrong credentials and once the request is captured in the Proxy tab, click HTTP history sub-tab of Proxy, select the appropriate POST request, right click and choose "Send to Repeater. Alternatively, you can do the same from Target tab as well. Note that the Repeater tab will turn orange.
- Now go to the Repeater tab, and you will notice that the request we just sent will appear in the left pane, but without any Response from server as shown in the screenshot below:



Step 2: Modify the Request and Send

- Inside the Repeater tab, modify the `username:password` and click the Send button and analyze how the modifications affected the server response.
- Manually test multiple username-password combinations in repeater, and in the next handout we will see automated tools like Intruder.
- The advantage of using Repeater tab over Proxy tab is that it allows persistent editing of requests without needing the application to re-trigger them and it is easier to test multiple variations of a single request.

To Do:

Practice the use of Repeater tab on the following websites:

- **http://<IP of M2>/basicloginapp1/index.html**
- **http://<IP of M2>/basicloginapp2/index.html**
- **https://juice-shop.herokuapp.com/**