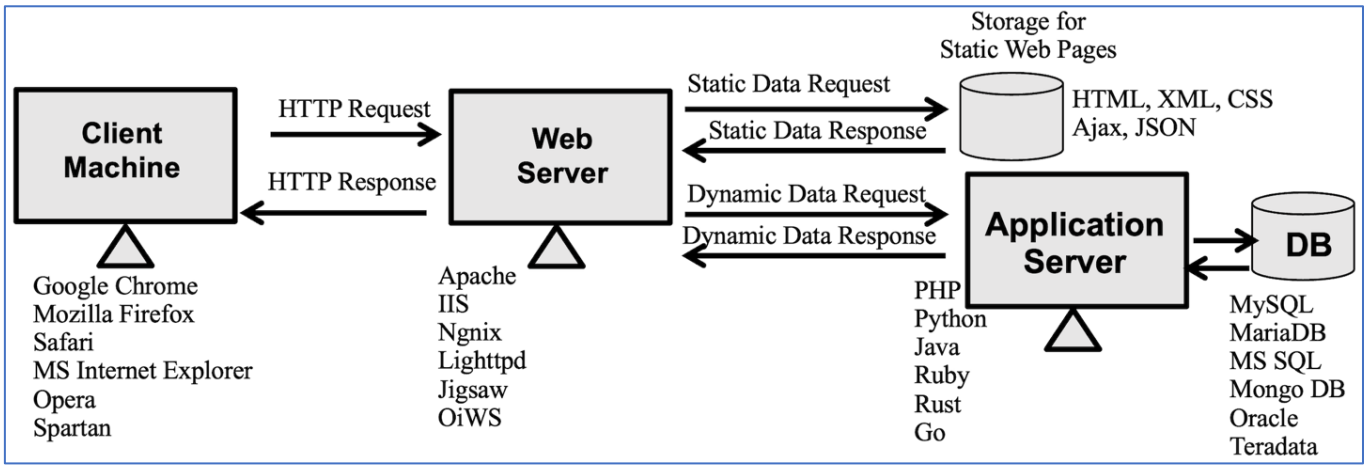
HO#2.9 Web App Penetration Testing - I

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 to find the vulnerable services running on different Linux and Windows based machines and exploiting them.

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 application's logic, input validation, authentication and APIs using tools like* `Burp Suite`, `OWASP ZAP`, `Nikto`, `MSF`, `BeEF`, `Hydra`, `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 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:

- HTTP (80): For unsecured website traffic.
- HTTPS (443): For secured website traffic.
- SMTP (25), and POP3 (110): For sending and receiving email by organizations.
- Port 21 (FTP): For transferring files to and from the servers.

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

The weaknesses or flaws in web-based applications are 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** (https://owasp.org/www-project-top-ten/) 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- Server-Side Request Forgery

- **Broken Access Control:** When users can do things they're not authorized to, like accessing other users' data or performing admin actions without permission.
- **Cryptographic Failures:** Sensitive data isn't protected properly, like passwords or credit card numbers being transmitted or stored without encryption, making them easy to steal.
- **Injection:** Malicious data is sent to a program (like shell commands, code, or SQL queries) to trick it into doing something harmful, such as leaking sensitive data or taking control of the entire system.
- **Insecure Design:** The application is built in a way that doesn't consider security from the start, making it vulnerable to attacks.
- **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 credentials/settings.
- **Vulnerable and Outdated Components:** Using old or vulnerable libraries or software in your application, which attackers can use to break in or exploit weaknesses.
- **Identification and Authentication Failures:** This refers to weaknesses in the identification and authentication mechanisms that can lead to unauthorized access. For example, use of weak passwords or lack of Multi-Factor-Authentication (MFA).
- **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.
- **Security Logging and Monitoring Failures:** The application or system doesn't keep track of important security events (like login attempts) or responding to attacks in real-time, allowing attackers to go unnoticed.
- **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.

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 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.

Tools used in Different Phases of Web-app Pen-Testing

- Reconnaissance: `Google Dorking`, `whois`, `nslookup`, `theHarvester`, `shodan`, `whatweb`, `wafw00f` and so on.
- Scanning and Vulnerability Analysis: `burpsuite`, `dirb`, `nikto`, `nmap`, and so on.
- Exploitation: `burpsuite`, `MSF`, `BeEF`, `hydra`, `john`, `medusa`, `SQLMap` and so on.
- Post Exploitation: `burpsuite`, `MSF`, `BeEF` and so on.

Dear students, we will be approaching the Top Ten OWASP vulnerabilities one by one, and will try to answer the following questions with practical hands-on examples for each:

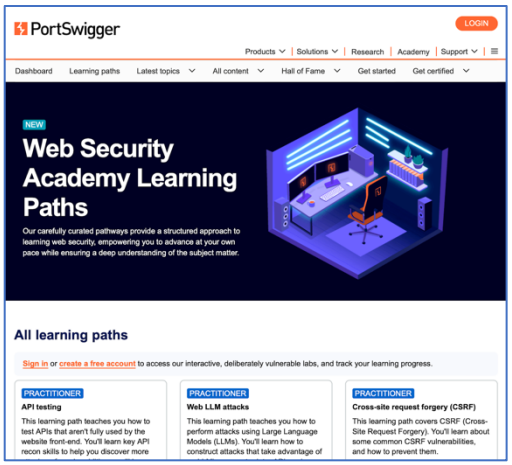
- **What is AX Vulnerability?**
- **How do you find an App suffers with AX Vulnerability?**
- **How do you exploit AX Vulnerability?**
- **How do you prevent/mitigate AX Vulnerability?**

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. (While running these vulnerable programs, your machine will be extremely vulnerable to attacks, so better is to disconnect from the Internet while using them):
  - **Damn Vulnerable Web Application:** https://github.com/digininja/DVWA
    - Frontend: HTML, CSS, JavaScript
    - Backend: PHP (Hypertext Preprocessor) 5.2.4-2
    - Database: MySQL 5.0.51a
    - Webserver: Apache 2.2.8
  - **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 as well like MongoDB
    - Webserver: Built-in Node.js server
    - Online instance is available at https://juice-shop.herokuapp.com

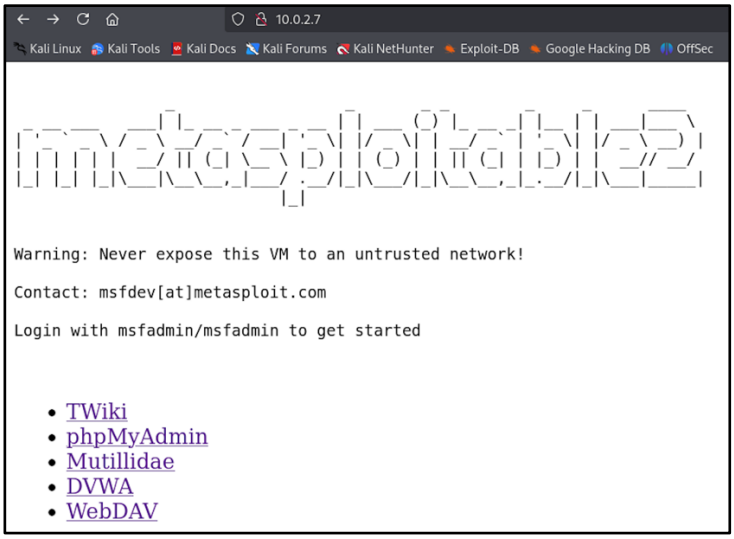
- **Option 2:** https://portswigger.net/web-security/learning-paths
  Create an account on *PortSwigger Web Security Academy*, which is a free online training platform created by PortSwigger, the developers of Burp Suite. It is designed to help individuals learn web application security concepts, vulnerabilities, 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.

Vulnerable Web Applications on M2

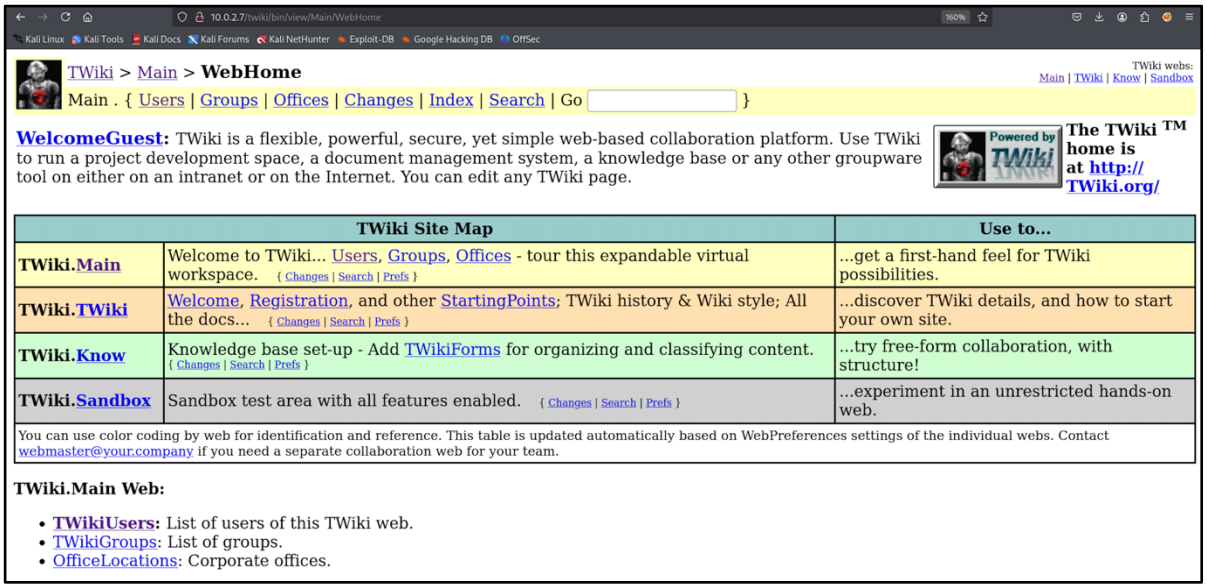
- The Metasploitable2 (M2) machine has lot of vulnerable Web applications that can be accessed from Kali machine via http://<M2-IP>/index.php. In the screenshot below, you can see five different links that can be used to practice and exploit different vulnerable web applications, although we will mainly be dealing with DIWA.



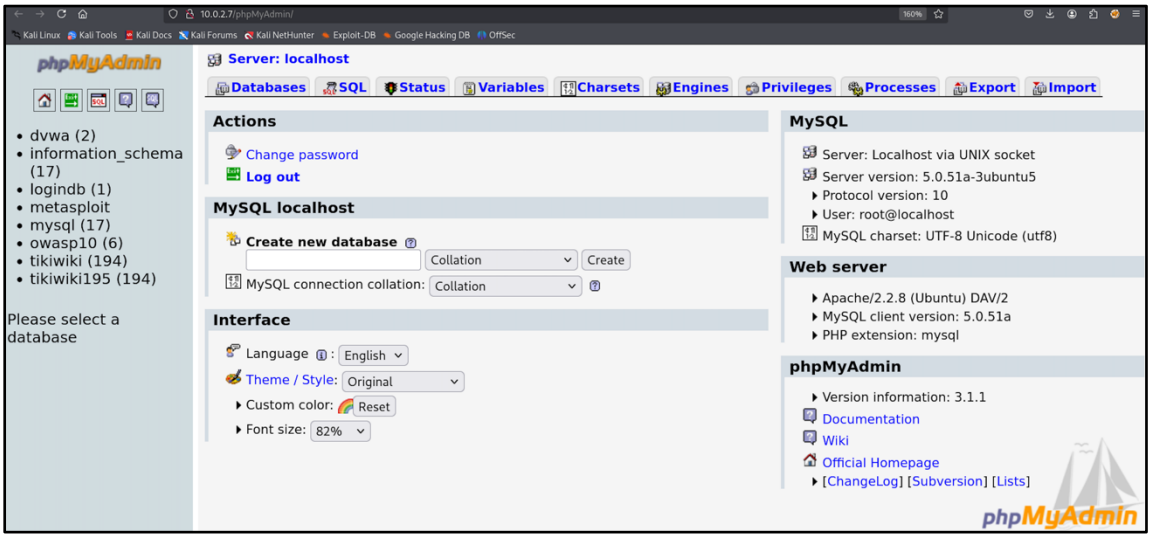
- From Kali, do a remote login on M2 and check out the contents of `/var/www/` directory of M2:

```
kali@kali:~$ ssh -ohostkeyalgorithms=ssh-dss msfadmin@<M2-IP>
msfadmin@metasploitable:~$ ls /var/www
index.php       dvwa/       tikiwiki/          phpMyAdmin/   mutillidae/   dav/
books/          sqli/       basicloginapp1/    basicloginapp1/
```

TWiki: A *web-based collaboration platform* that allows users to easily add, edit, and update content. The most famous example is Wikipedia used for creating and managing wikis, enabling team collaboration and document sharing. (http://<M2-IP>/twiki/index.html)

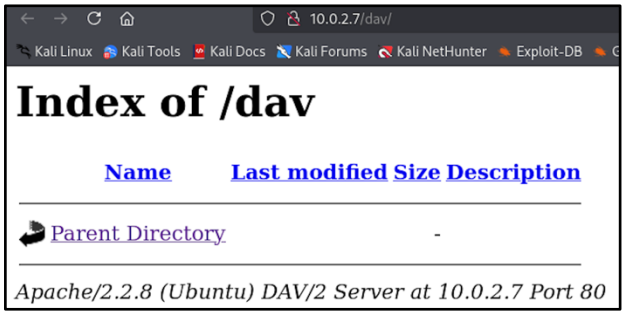


phpMyAdmin: A popular open-source tool written in PHP for managing MySQL or MariaDB databases through a web interface with credentials `root:blank/123456`. (http://<M2-IP>/phpMyAdmin/index.php)

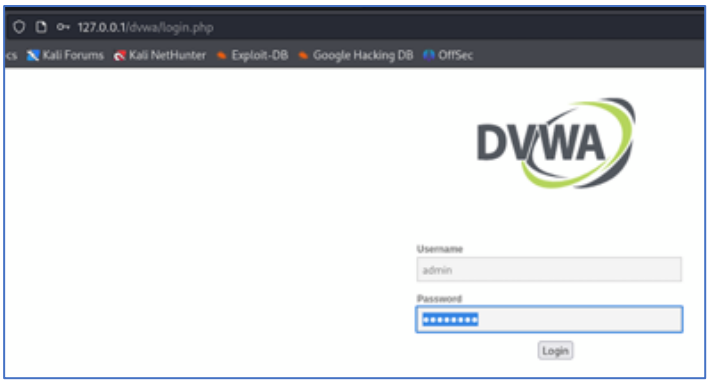


Mutillidae: A deliberately vulnerable web application used for practicing web application security testing, such as penetration testing and ethical hacking. (http://<M2-IP>/mutillidae/index.php)

WebDAV: A protocol that extends HTTP to enable the management of files on remote servers, allowing clients to upload, download, and manage files on a web server. This is a shared directory `/var/www/dav/` on Metasploitable 2 with its sticky bit set.



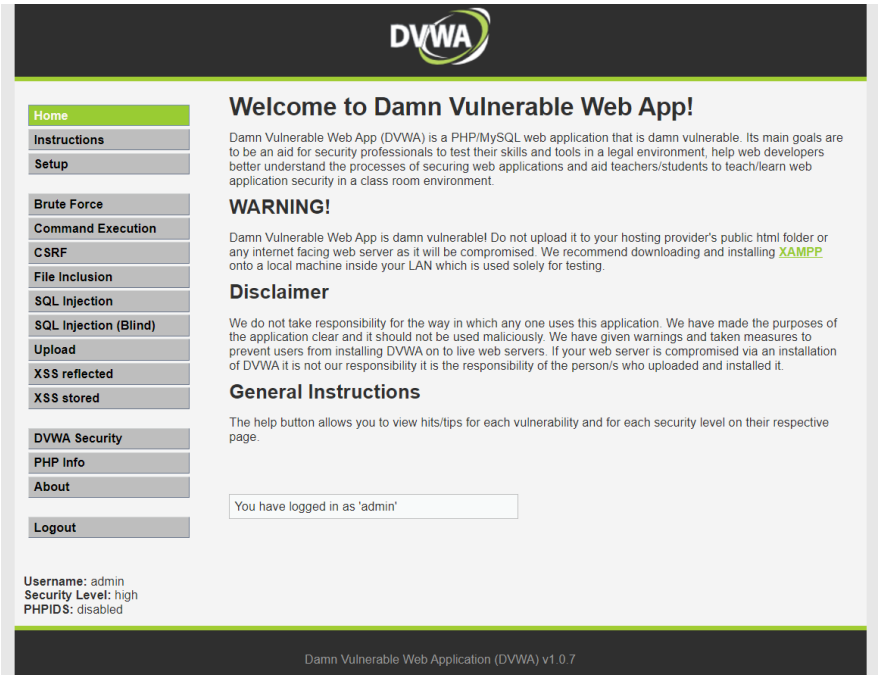
Lab Environment for DVWA on M2



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 cracking common web vulnerabilities, all within a simple, straightforward interface. From Kali Linux visit http://M2-IP/dvwa/login.php to access its login page as shown in the opposite screenshot and login using the credentials (`admin:password`), and it will take you to http://M2-IP/dvwa/index.php page, as shown in the following screenshot.

In the left pane, you can see that there are different buttons, which are links to vulnerable pages corresponding to different attacks like Brute Force, Command Injection, SQLi, XSS, CSRF and so on. Moreover, by clicking the DVWA Security button in the left panel, you can set pen testing difficulty level of DVWA to a level of your choice:

- **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.



Before you start, please click the **Instructions** button in the left pane, and give a bird's eye view to the page contents, which might be helpful, if you want to install and setup DVWA and its Database on local your Kali Linux machine.

Problem:

For the first time you may get a message "*Unable to connect to the database Click here to setup the database*". If you see this problem perform the following steps:

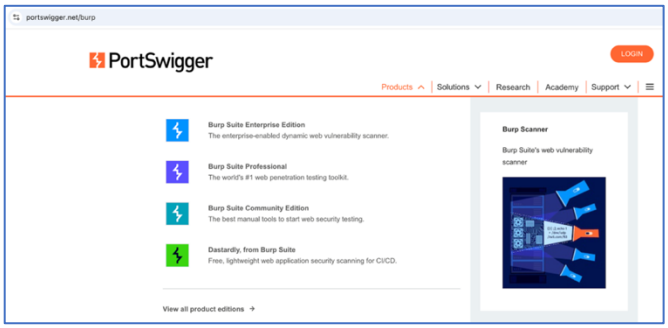
- On M2, edit the `/var/www/dvwa/config/config.inc.php` file and change the `db_server` from `'localhost'` to `'127.0.0.1'` and set the password appropriately. Then you need to restart `mysql` by giving `sudo /etc/init.d/mysql restart` command.
- On M2, edit the `/etc/php5/apache2/php.ini` file and change the `allow_url_fopen` and `allow_url_include` paramters to **On**. Then you need to restart `apache2` by giving `sudo /etc/init.d/apache2 restart` command.

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 server 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 suffering with BoF vulnerability, which the application then executes as part of its normal flow.
  - **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.
  - **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 a complete access to db 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. RFI is a more dangerous variation, where the attacker provides a URL to a file located on a remote server. If the application fails to validate the input properly, the attacker can include and execute malicious scripts leading to remote code execution (RCE).
- **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. Attacker 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 (stored inside cookies).
  - **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)**: CSRF is a type of attack where an attacker tricks a user into performing actions on a web application where they are authenticated, without their consent. This often leads to actions like changing passwords or making requests on their behalf. An example of such attack is the attacker sends the user a link that when clicked, changes the user's official email address or phone number on a banking site without their consent.
- **Server-Side Request Forgery (SSRF):** SSRF is a vulnerability where an attacker can trick an application into making requests to other servers, allowing attackers to access sensitive information or interact with systems that should be off-limits.
- **Password Hashing**: Exploit weak password hashing algorithms (e.g., MD5, SHA1) by cracking hashed passwords.

Note: 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. ☺

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 is used to capture and analyze the packets (TCP, UDP, HTTP etc.) flowing through a network in real-time. Now we are going to study another tool called **`Burp Suite`**, which is a web application penetration testing framework developed by PortSwigger (https://portswigger.net/burp). It allows a user to intercept, inspect, and modify HTTP/S traffic in real-time 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. If you get an error when running `burpsuite`, you may need to select the appropriate version of Java RunTime (`/usr/lib/jvm/java21-openjdk-amd64/bin/java`) using following command:

Check for installed JDKs:

```
┌──(kali㉿kali)-[~]
└─$ update-java-alternatives --list
java-1.11.0-openjdk-amd64       1111       /usr/lib/jvm/java-1.11.0-openjdk-amd64
java-1.17.0-openjdk-amd64       1711       /usr/lib/jvm/java-1.17.0-openjdk-amd64
java-1.21.0-openjdk-amd64       2111       /usr/lib/jvm/java-1.21.0-openjdk-amd64
```

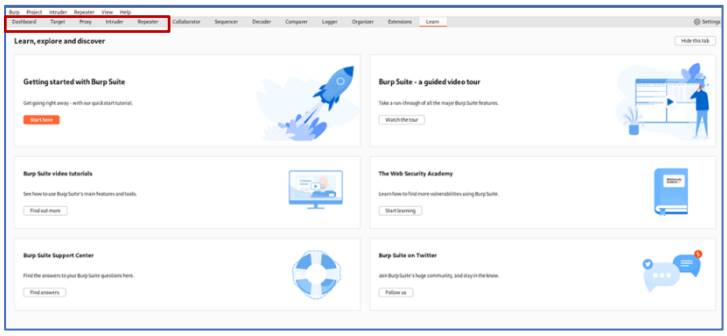
Switch to appropriate JDK: Run the following command for `java` as well as for `javac`

```
┌──(kali㉿kali)-[~]
└─$ sudo update-alternatives --config java
There are 3 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-21-openjdk-amd64/bin/java      2111      auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      manual mode
  2            /usr/lib/jvm/java-17-openjdk-amd64/bin/java      1711      manual mode
* 3            /usr/lib/jvm/java-21-openjdk-amd64/bin/java      2111      manual mode

Press <enter> to keep the current choice[*], or type selection number: 3
```

Verify and Check the current version of `java` and `javac`

```
┌──(kali㉿kali)-[~]
└─$ java --version
openjdk 21.0.7-ea 2025-04-15
OpenJDK Runtime Environment (build 21.0.7-ea+7-Debian-1)
OpenJDK 64-Bit Server VM (build 21.0.7-ea+7-Debian-1, mixed mode, sharing)
```

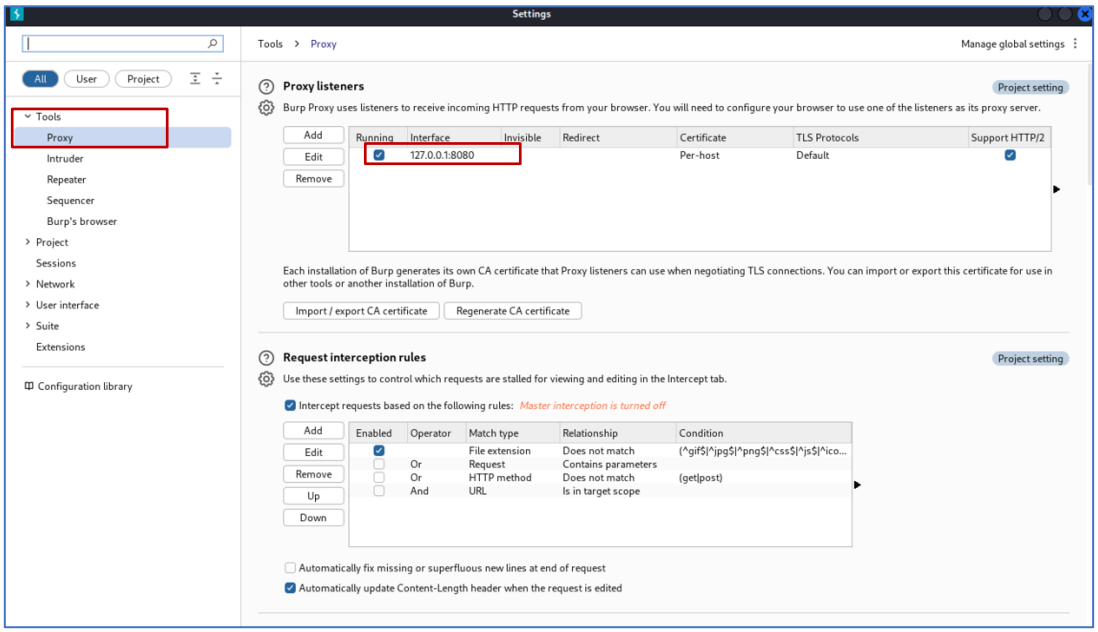
Note: If JAVA_HOME variable is set then it takes priority on above settings.



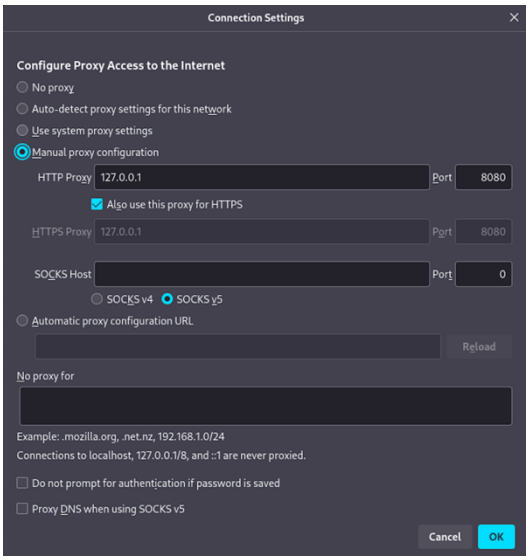
Using `Burp`, we can *analyze* HTTP requests, can *modify* them and *resend* the modified requests. Being able to edit requests that our browser sends is the key to testing a web application. So, this is the core thing (Intercept) that `Burp` do, other than many other fancy tasks like Repeater (repeats requests), Intruder (brute force tools) etc. The opposite screenshot is the main screen of Burp Suite, where you can see different tabs ***Target, Proxy, Repeater, Intruder,*** *Fuzzer* and so on depending on whether you have Enterprise, Professional or Community Edition of Burp Suite. The *Fuzzer* tool (not available in community edition) generates large volumes of random or malformed inputs to a specific parameter or input field in a web application to identify vulnerabilities like buffer overflows, SQL injection, cross-site scripting (XSS) etc. The *fuzzer* can also help stress-test an application by sending high volumes of traffic or inputs that could overwhelm the application, leading to crashes or unexpected behavior).

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. Once done, close the window:



- **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, fire up Firefox inside Kali Linux, open **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. 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.



- Once done practicing with Burp, i.e., you do not want Burp to capture HTTP/S traffic anymore, you must turn off the proxy settings inside your browser.

- **Note:** Firefox proxy settings don't affect traffic outside the browser (e.g., terminal commands). If you want to route all traffic through Burp Suite, including terminal commands, you need to configure a system-wide proxy. On Linux this can be done by setting the `http_proxy` and `https_proxy` environment variables using following commands:

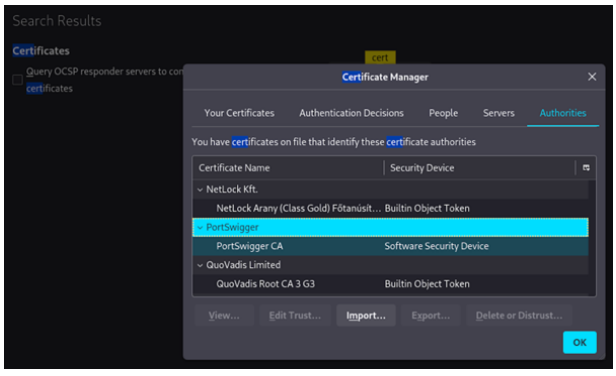
```
$ export http_proxy="http://localhost:8080"
$ export https_proxy="http://localhost:8080"
```

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, you need to install/import the Port Swigger's Certificate in your browser (Firefox).
  - **What is CA certificate:** A CA (Certificate Authority) certificate is a digital certificate issued by trusted organizations like GoDaddy, DigiCert, and GlobalSign. 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 your Kali Linux Download directory. You can use the following command to view the public key and signature along with other certificate details (private key is not part of the certificate):

```
$ openssl x509 -inform DER -in cacert.der -noout -text
```

  - **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".

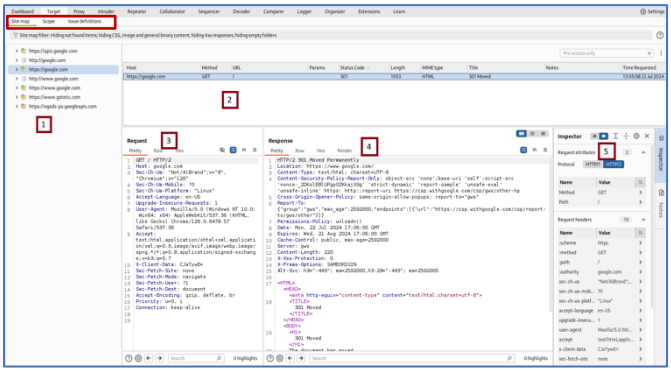


  - Now try intercepting HTTPS traffic inside Burp by visiting some https web site say our favorite https://arifbutt.me. This will work now, to verify just go on the Target tab of Burp, where you can see the incoming and outgoing HTTP request/response messages ☺

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 all the websites that you are visiting via your Firefox browser. The captured information is divided into five panes:

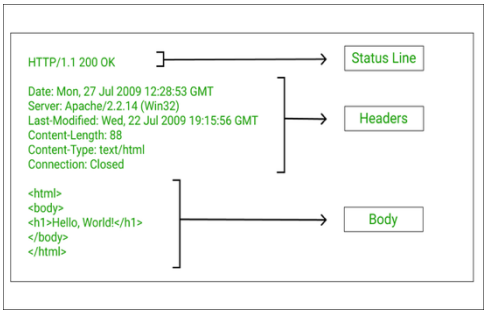
1. On the left the **Site map** pane displays a hierarchical tree of the target application, showing directories (folder icons), files (document icon), parameterized requests (gear icon) and individual URLs that make up the site. As you browse or scan, Burp Suite dynamically update the site map with 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.
- **Site Map Filter** helps you focus on specific parts of your web application during testing. Click on the Site Map Filter and choose what to filter and what not to filter.
- Issue **definitions sub-tab** list known vulnerabilities and security issues with explanations and guidance for fixing them.

Hands On Practice:

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

- From Kali Linux, visit our basic website **http://<M2-IP>/basicloginapp1/index.html** deployed on M2, and send the credentials to login (`arif:kakamanna`). It is using the **GET** method to send the form parameters (inside the URL `?var1=value1&var2=value2…`). The limitations of using the HTTP GET method: Form data is appended to the URL in clear, and limited to typically 2048 characters in modern browsers.

- Visit our basic website **http://<M2-IP>/basicloginapp2/index.html** deployed on M2, 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. The sample code can be viewed on M2 inside `/var/www/basicloginapp2/index.html` and `login.php` files. (Screen shot shown below):

```
//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:

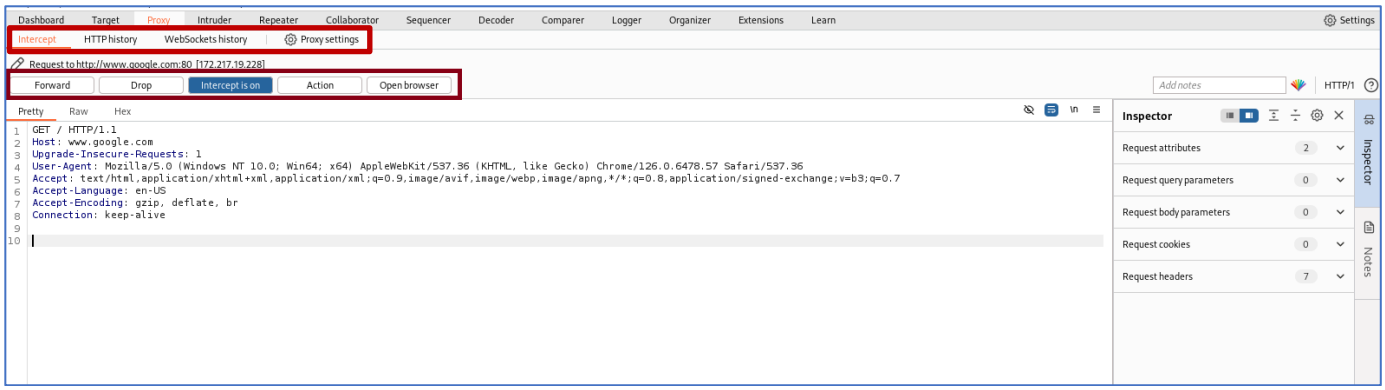
Login

```
//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 the DVWA website **http://<IM2-IP>/dvwa/login.php** deployed on M2 machine, and send the credentials to login (`admin:password`). It is using the POST method to send the form parameters. Do note on giving incorrect credentials, the HTTP Response (in Burp) show Location:`login.php`, while giving correct credentials, the HTTP Response (in Burp) show Location:`index.php`, and on the browser as well you move from **login.php** to **index.php** page of the website.

Proxy Tab of Burp Suite: *The Proxy tab with **Intercept turned ON** is used for real time inspection and modification of HTTP requests or responses before forwarding them to the web server or to the client browser respectively.* 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.

- 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.

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:

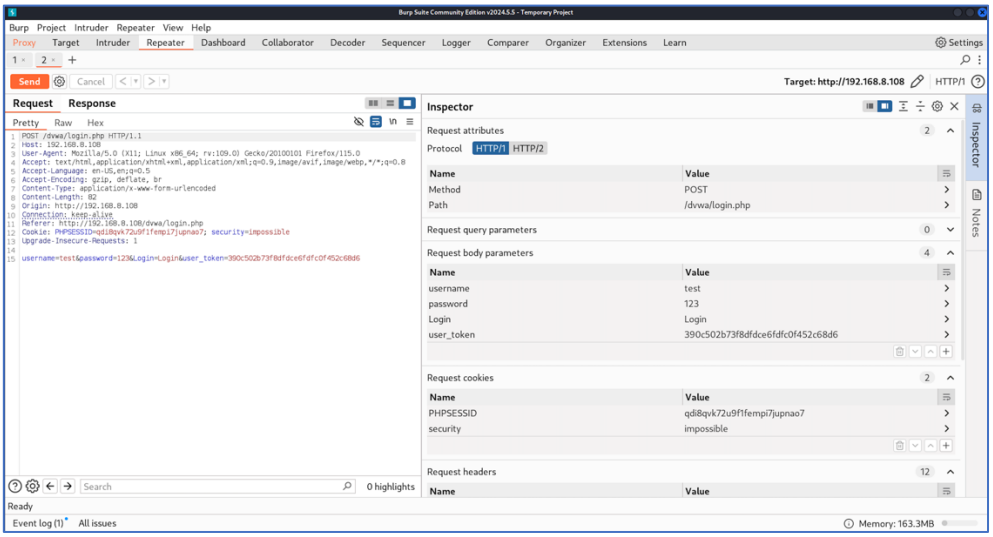
- **http://<IP of M2>/basicloginapp1/index.html**
- **http://<IP of M2>/basicloginapp2/index.html**
- **http://<IP of M2>/dvwa/login.php**

Repeater Tab of Burp Suite: The *Repeater* tab is used to manually modify and resend individual HTTP/S requests (header, body, method, parameters, cookies etc.) repeatedly to the server and then analyze server's response to identify vulnerabilities. It is useful for:

- Parameter tampering.
- Identifying vulnerabilities such as SQLi, XSS, etc.
- Experimenting with authentication or session management.
- Input fuzzing.

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 later we will see as to how the Intruder tab automate this process.
- 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.

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/**

A7: Identification and Authentication Failures

In the **OWASP Top 10** (https://owasp.org/www-project-top-ten/) list of 2021, *Identification and Authentication Failures is at level 7*. This refers to weaknesses in the identification and authentication mechanisms that can lead to unauthorized access. For example, use of weak passwords or lack of MFA. Before we proceed and perform attacks on web apps that suffer with this weakness, let us first revise three related concepts, which are ***Authentication***, ***Session Management*** and ***Access Control***.

- **Authentication:** Websites are potentially exposed to anyone who is connected to the Internet. Authentication is the process of verifying the identity of a user or client. In order to access services on a website, a user normally enters his/her credentials, which are sent to the server for verification. A user can be authenticated using a combination of following techniques:
    - Something you know (password, passphrase, PIN)
    - Something you have (NIC, ATM, Passport)
    - Something you are (Biometrics)
    - Somewhere you are (Geographic location, IP, MAC address)
    - Something you do (signatures, pattern unlock)
- **Session Management:** We all know that HTTP(S) protocol is stateless, however, a good web application needs to make it stateful. For example, once a user has been authenticated by the server, the user may like to visit other pages of the application, e.g., his/her profile page, or change password page and so on. Now all these pages are authenticated pages, so one way is that the user has to give his/her `username:password` every time he/she wants to visit an authenticated page and the other way is using Session ID. ***Session Management is a process that involves maintaining state between a client's browser and the web application on the server across multiple requests, as HTTP itself is a stateless protocol***. Here is a breakdown of how session management is done at an abstract level:
    - ***Session Initialization:*** A user is authenticated by a login form, and upon successful authentication the server creates a unique session ID, which is sent back to the client and is stored in cookies (a small piece of alphanumeric data sent by the server and stored on the client's browser), URL parameters or HTML hidden fields. (Think of a SID as your visitor badge number, and a cookie as the plastic badge holding it).
    - ***Session Continuation:*** The Session ID gets passed by the browser to the server with every request that the user make and all of this happens in the background. The server uses this Session ID to retrieve the user's session data from its memory or a database.
    - ***Session Termination:*** The session ends when either the user logs out, or due to inactivity by the user. Two related types of cookies in this context are:
        - Persistent Cookies are stored on the user's device even after the browser is closed. They remain valid until their expiration date or until the user manually deletes them. (`document.cookie = "user=arif; expires=Sun, 28 Jul 2025 12:00:00 UTC";`)
        - Non-persistent cookies, also called session cookies, are stored temporarily in the browser's memory and are deleted when the browser is closed. (`document.cookie = "user=arif";`)
- **Access Control:** Access Control determines whether the user is allowed to carry out the actions that he/she is attempting to perform. There are different access control mechanisms like Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC), to name a few. *Broken Access Control (A1)* is a critical web application security vulnerability that occurs when an application does not properly enforce restrictions on what authenticated or unauthenticated users can access or perform. Users can do things they're not authorized to, like accessing other users' data or performing admin actions without permission.

Vulnerabilities in Password Based Logins:

- A *brute-force attack* is when an attacker uses a system of trial and error to guess valid user credentials. Brute-forcing is not always just a case of making completely random guesses at usernames and passwords. Attackers normally use publicly available knowledge to fine-tune brute-force attacks and make much more educated guesses (dictionary attacks, use of rainbow tables). These attacks are typically automated using wordlists of usernames and passwords, using dedicated tools like `hydra`, `medusa`, and `burp suite`.
- A *username enumeration* is when an attacker is able to observe changes in the website's behaviour in order to identify whether a given username is valid. Username enumeration typically occurs on a login page, when you enter a valid username but an incorrect password and get a specific message. It can also occur on signup/registration forms when you enter a username that is already taken. This greatly reduces the time and effort required to brute-force a login because the attacker is able to quickly generate a shortlist of valid usernames.

Vulnerabilities in Multi-Factor Authentication:

- Many websites rely exclusively on single-factor authentication using a password to authenticate users. However, some require users to prove their identity using multiple authentication factors. Verifying biometric factors is impractical for most websites, however, most websites use two-Factor Authentication (2FA) based on something you know and something you have. This usually requires users to enter both a traditional password and a *temporary verification code sent to an out-of-band physical device in their possession*. It is sometimes possible for an attacker to obtain a single knowledge-based factor, such as a password, but simultaneously obtaining another factor from an out-of-band source is considerably less likely. For this reason, two-factor authentication is demonstrably more secure than single-factor authentication.
- Poorly implemented two-factor authentication can be beaten/bypassed, just as single-factor authentication can. For example, if the user is first prompted to enter a password, and then prompted to enter a verification code on a separate page, the user is effectively in a "logged in" state before they have entered the verification code. In this case, it is worth testing to see if you can directly skip to "logged-in only" pages after completing the first authentication step. Occasionally, you will find that a website doesn't actually check whether or not you completed the second step before loading the page.

Vulnerabilities in Other Related Services:

- **Password Reset Functionality:** The mechanisms of changing or resetting a user password can also introduce vulnerabilities that can be exploited by an attacker. Developers usually take care to avoid well-known vulnerabilities in their login pages, but they overlook the fact that they need to take similar steps to ensure that related functionality is equally robust as the login page.
- **Keeping Users Logged In:** A common feature that exist in websites is the option to stay logged in even after closing a browser session. This functionality is often implemented by generating a "remember me" or "Keep me logged in" token of some kind, which is then stored in a persistent cookie. As possessing this cookie effectively allows you to bypass the entire login process, it is best practice for this cookie to be impractical to guess. However, some websites generate this cookie based on a predictable concatenation of static values, such as the username and a timestamp. Some even use the password as part of the cookie. This approach is particularly dangerous if an attacker is able to create his own account, and then by study his/her own cookie can deduce how it is generated. Once they work out the formula, they can try to brute-force other users' cookies to gain access to their accounts.

Preventing Authentication Vulnerabilities:

If an attacker is able to find flaws in an authentication mechanism, they would then successfully gain access to other users' accounts, allowing the attacker to access sensitive data. *Following are the key points that web developers must follow in order to prevent authentication vulnerabilities:*

1. Use strong authentication mechanisms:
    - Implement MFA.
    - Avoid weak passwords.
    - Ensure that application uses an encrypted HTTPS connection to transmit login credentials.
2. Secure password storage:
    - Store passwords using strong hash algorithms like bcrypt, scrypt, Argon2, instead of md5, sha-1, and sha-256.
    - Use salting to avoid rainbow table attacks.
3. Prevent credential stuffing:
    - Limit the number of login-attempts per IP or account by locking accounts temporarily after a certain number of failed login attempts.
    - Use CAPTCHAs to distinguish between human and automated login attempts.
4. Implement logging of failed login attempts and keep monitoring them.
5. Change all default credentials.

Offline vs Online Password Attacking Tools:

- **Offline Password Attacks:** Offline password attacks typically involve obtaining password hashes or files from compromised systems or databases (Windows SAM or Linux `/etc/shadow` file). Once the attacker has these hashes, they can use specialized software and hardware to crack the passwords at an accelerated rate (requires high performance systems). Commonly used tools for offline password attacks are `hashcat`, `John the Ripper`, `Cain and Abel`.
- **Online Password Attacks:** Online password attacks occur in real-time and directly target the authentication process, i.e., a web form asking username and password. These attacks are very noisy, as failed login attempts are logged into the server. Online attacks face additional challenges, e.g., you may get locked out after a certain number of failed login attempts, CAPTCHA and rate-limiting mechanisms to detect and prevent automated attacks. Commonly used tools for online password attacks are `hydra`, `medusa`. `Aircrack-ng` is a hybrid tools that uses both online and offline phases in order to crack WEP and WPA/WPA2 Wi-Fi passwords.
- **Generating Wordlists using `crunch`:** The password cracking tools need wordlists which you can download from the Internet, or can use existing wordlists inside your Kali machine located in the `/usr/share/wordlists/` directory. Another option is creating your own wordlist using tools like **`crunch`** as shown below:

```
# crunch <min-len> <max-len> [<charset string>] -o <filename>
```

```
# crunch 4 4 ab -o mywordlist.txt    (2^4 = 16)
# crunch 2 4 ab -o mywordlist.txt    (2^2 + 2^3 + 2^4 = 4 + 8 + 16 = 28)
# crunch 2 4 abc -o mywordlist.txt   (3^2 + 3^3 + 3^4 = 9 + 27 + 81 = 117)
# crunch 3 6 abcd -o mywordlist.txt  (4^3 + 4^4 + 4^5 + 4^6 = 64 + 256 + 1024 + 4096 = 5440)
# crunch 8 8 abc…z -o mywordlist.txt (26^8 = 208827064576)
```

Imagine the size of the wordlist containing exactly 8-xter passwords containing lower+upper+digits+special characters ☺

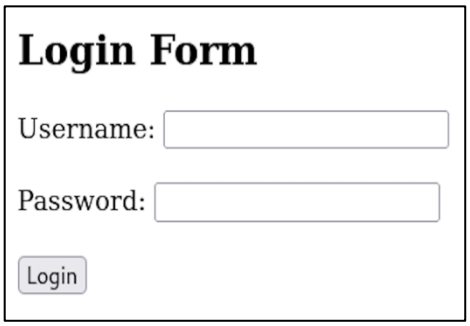
Example 1: Brute Force Attack on a Basic Login App

Let us suppose we have a basic web site consisting of just two files `index.html` and `login.php`, which are there inside the `/var/www/basicloginapp2/` directory on our M2 Linux machine.

```
//index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <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><br><br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required><br><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

```
//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.";
?>
```

Start Apache2 service on M2 and access this `basicloginapp` by opening a browser on Kali and typing the address `http://<IP of M2>/basicloginapp2/index.html`. This will display the `index.html` page as shown. Use Burp, try giving different credentials and see how the HTTP request object is sent to the `login.php` file on the server, and depending on the credentials, the server either returns a string "*Login successful*" or "*Invalid username or password*".



Brute Force Attack on basicloginapp2 using hydra:

The ***hydra*** is a popular and powerful, command line based online password-cracking tool designed to perform brute-force attacks against login credentials for various network protocols, services, and applications. It is widely used by penetration testers and security researchers to test the strength of authentication mechanisms.
Before using automated tools to launch a Brute Force attack, we need to create two text files one containing usernames and the other containing passwords manually or using `crunch`. Alternatively, you can use some existing files in Kali or may download some freely available from the Internet. Let us use following files:

~/usernames.txt	~/passwords.txt
admin root msfadmin arif	kakamanna msfadmin password

Another thing you need to check out are the parameter names which we want to brute force by opening this link `http://<M2 IP>/basicloginapp2/index.html` in your Kali browser and view source. The names of the parameters in this case are `username`, `password` and `Login`.

```
$ hydra -L ~/usernames.txt -P ~/passwords.txt <IP of M2> http-post-form
"/basicloginapp2/login.php:username=^USER^&password=^PASS^&Login=Login:Invali
d username or password"
```

- **-L** option specifies the path to your username list.
- **-P** option specifies the path to your password list.
- **<IP of M2>** specifies the IP address of server.
- **http-post-form** followed by a string mentioning *path*, *form parameters* , optional *cookie value* and the *failure string* (all arguments separated by colons).

```
┌──(kali㉿kali)-[~]
└─$ hydra -L ~/usernames.txt -P ~/passwords.txt 192.168.8.108 http-post-form "/basicloginapp/login.php:username
=^USER^&password=^PASS^:Invalid username or password"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organi
zations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-19 12:35:16
[DATA] max 12 tasks per 1 server, overall 12 tasks, 12 login tries (l:4/p:3), ~1 try per task
[DATA] attacking http-post-form://192.168.8.108:80/basicloginapp/login.php:username=^USER^&password=^PASS^:Inva
lid username or password
[80][http-post-form] host: 192.168.8.108   login: arif   password: kakamanna
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-19 12:35:16
```

From the output of command, we see that hydra tried 12 combinations, and finally have given us the correct credentials `arif:kakamanna` using which we can now successfully login ☺

Brute Force Attack on `basicloginapp2` using Burp Suite:

The ***Intruder* tab** of Burp Suite uses the same concepts as Repeater, but adds automation and complexity. It is used to perform customized and automated attacks on specific parameters or inputs of a web request. Common use cases include *brute-forcing*, *SQL injection*, *Cross-Site Scripting (XSS)* and *parameter fuzzing*.

Step 1: Send appropriate Request Object to Intruder:

- Start browser, and enter the address `http://<M2 IP>/basicloginapp2/index.html` in Kali browser and view source. The names of the parameters in this case are `username`, `password` and `Login`.
- Start Burp Suite, and under the Proxy tab make the Intercept OFF
- On the browser, give wrong credentials `abc:xyz` and click Login button.
- On Burp Target tab, select the appropriate URL with POST method, and check out the request and response objects. Right click on the Request object and press "*Send to intruder*".

Step 2: Select Attack Type and Add Payload positions:

- The Intruder Tab has further four sub-tabs *Positions, Payloads, Resource pool,* and *Settings*.
- Select the Positions sub-tab and **choose Attack type:**, out of the following four:

  Payload1 (username) = [arif, kakamanna, rauf]

  Payload2 (password) = [fcit, mit, fcu]

  - **Sniper:** One position is attacked at a time; the others stay constant. Used in fuzzing, where you test one parameter at a time to see exactly which parameter is vulnerable.

```
username=arif&password=§pass§
username=kakamanna&password=§pass§
username=rauf&password=§pass§
username=§user§&password=pucit
username=§user§&password=mit
username=§user§&password=fcu
```

  - **Battering ram:** Same payload is used in multiple positions, so two lists are combined to a single list.

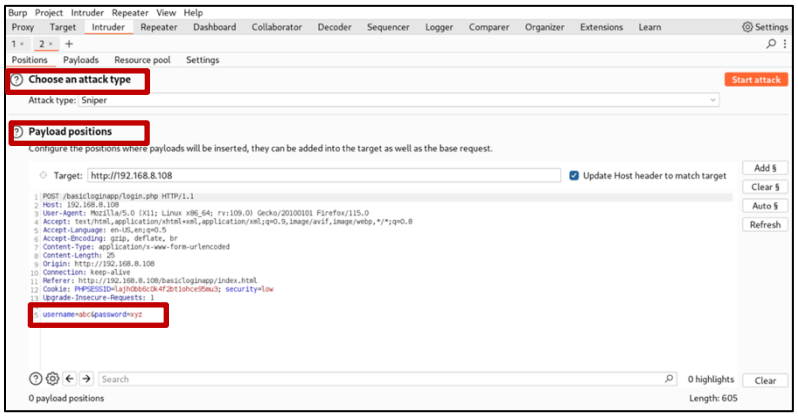
```
username=arif&password=arif
username=kakamanna&password=kakamanna
username=rauf&password=rauf
username=pucit&password=pucit
username=mit&password=mit
username=fcu&password=fcu
```

  - **Pitchfork:** Each request uses nth payload from each list, perfect if you want to try known username and password pairs from a data breach.

```
username=arif&password=pucit
username=kakamanna&password=mit
username=rauf&password=fcu
```

  - **Cluster bomb:** Tries all combinations of both lists, like a nested loop. (Used for Brute-Forcing)

```
username=arif&password=pucit
username=arif&password=mit
username=arif&password=fcu
username=kakamanna&password=pucit
username=kakamanna&password=mit
username=kakamanna&password=fcu
username=rauf&password=pucit
username=rauf&password=mit
username=rauf&password=fcu
```

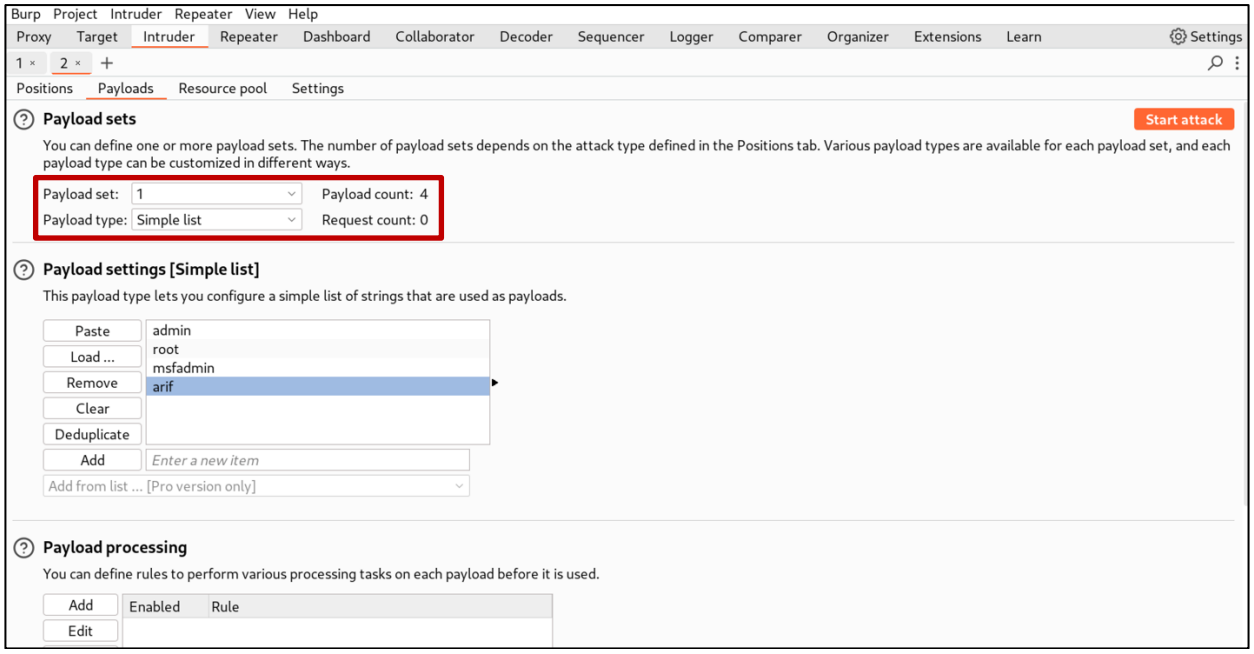


- Now from Positions sub-tab and **choose Payload positions**. In the Request object, first clear all the selected parameters (if any). Then select the user's name and click Add button. Repeat for the remaining parameters, i.e., password in this example.

Step 3: Select Payload sets and Add Payload Settings:

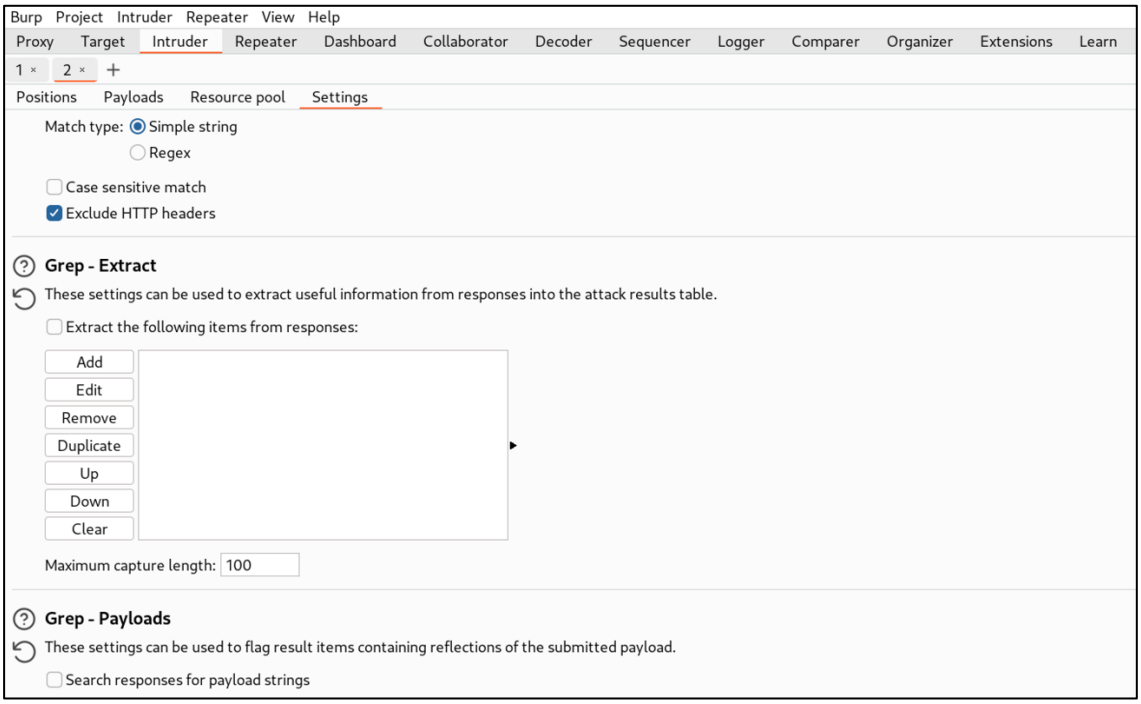
- Now click the Payloads sub-tab, and **choose Payload sets**, this will vary depending on the attack type you have selected in Step 2. Keep Payload type to Simple list. Now load Payload settings for each payload set. For Cluster Bomb attack type, you need to set two payloads
- Choose payload set 1 and add some usernames or load names from file.
- Choose payload set 2 and add some passwords or load passwords from file.

Note: Practice the usage of the four attack types to have a crystal-clear understanding of their differences.



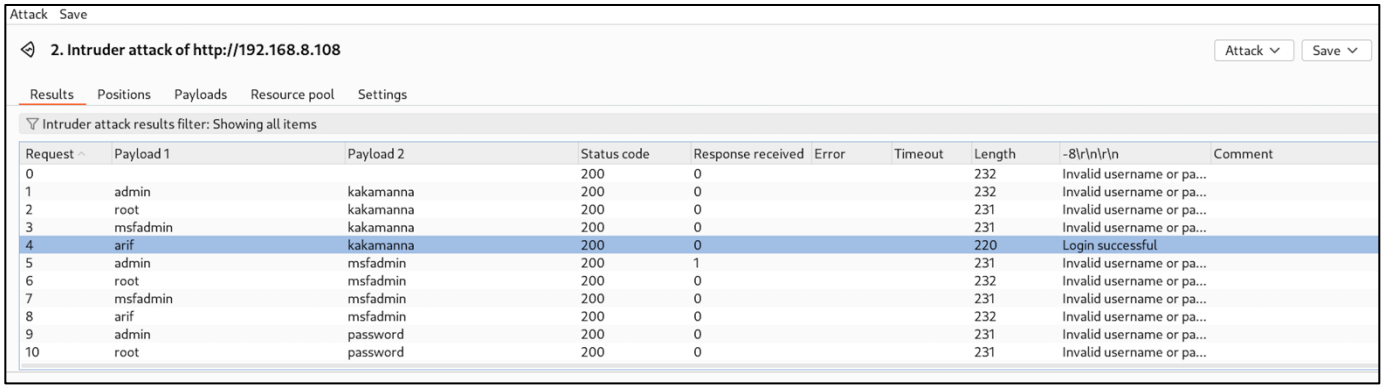
Step 4: Select Failure String from Settings sub-tab:

- Click Settings sub-tab, scroll to Grep-Extract, and click Add button.
- This will take you to a new window, select the Login failed string from the Response object and click OK.



Step 5: Launch Attack:

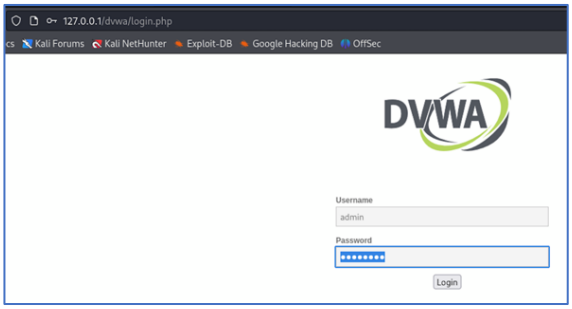
- Come back to Payloads sub-tab and click the Start Attack button at top right corner, and it will start the attack and show you the results as shown in the following screenshot. You can see the correct credentials which are `arif:kakamanna` ☺
- At times, you may not get a string that differentiate between the successful or failed login attempts. Then you have to analyze the data by checking the length. The different length (220) from others (232) indicates the correct username and password.

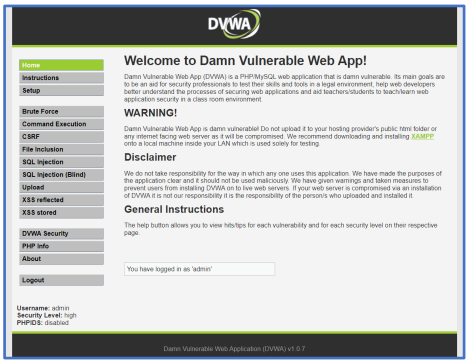


Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	-8\r\n\r\n	Comment
0			200	0			232	Invalid username or pa...	
1	admin	kakamanna	200	0			232	Invalid username or pa...	
2	root	kakamanna	200	0			231	Invalid username or pa...	
3	msfadmin	kakamanna	200	0			231	Invalid username or pa...	
4	arif	kakamanna	200	0			220	Login successful	
5	admin	msfadmin	200	1			231	Invalid username or pa...	
6	root	msfadmin	200	0			232	Invalid username or pa...	
7	msfadmin	msfadmin	200	0			231	Invalid username or pa...	
8	arif	msfadmin	200	0			232	Invalid username or pa...	
9	admin	password	200	0			231	Invalid username or pa...	
10	root	password	200	0			231	Invalid username or pa...	

To Do: Brute-Forcing DVWA Main Login Page using **hydra** and **burp**

http://<IP of M2>/dvwa/login.php





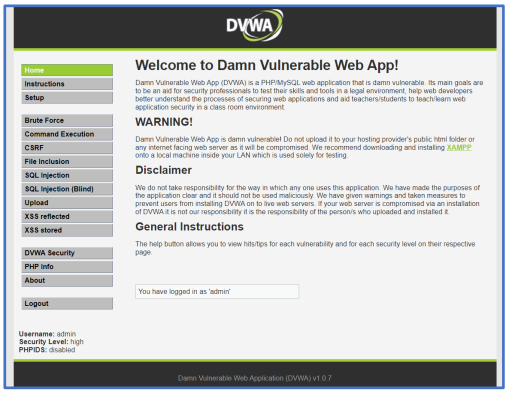
- Try giving different credentials manually, like `test:123`, `arif:pucit` and so on to finally the correct one, i.e., `admin:password` to understand how the application behaves on giving correct and incorrect credentials.
- Capture the login POST request inside Burp suite, and determine the POST parameters (e.g., `username=admin&password=12345&Login=Login`).
- Analyze the server's response after an unsuccessful login attempt and look for consistent text in the response that indicates failure. In DVWA login page the string **"Login failed"** is the failure string (the string to search in case of failure). If the failure string is not in the response, the tool might treat such attempts as successful. So, mention this string in the grep-extract.
- When you give the correct credentials `admin:password`, it will take you to the main page of DVWA having `http://<IP>/dvwa/index.php`, as shown

Hint:

- In this example, you may see the Login failed message on giving incorrect credentials, however, you may not find this string in the response object when using Burp. So, you may come across the issue of selecting the **Grep Extract** on the Settings tab of Intruder.
- If we look closely at the response messages, you will notice that for incorrect credentials response carries a key:value pair `Location:Login.php`, whereas for correct credentials `Location:index.php`
- So, in this example, when you go to the settings sub-tab of Intruder, you need to grep-extract and select `login.php` (the string to search in case of failure) and then click on Add button.
- This will work ☺

To Do: Brute-Forcing DVWA brute Login Page using **hydra** and **burp**

http://<IP>/dvwa/vulnerabilities/brute



- Login and access the DVWA application by opening a browser on Kali and typing http://<IP>/dvwa/login.php, giving the credentials admin:password, it will take you to the main page of DVWA having http://<IP>/dvwa/index.php. The main page of DVWA shows lot of vulnerable applications, as shown in the screenshot.

- Click the Brute Force button in the left pane and it will take you to http://<IP>/dvwa/vulnerabilities/brute address. Try giving different usernames and passwords manually and understand the behaviour of the application.

- Your task is to launch Brute Force attack on http://<IP>/dvwa/vulnerabilities/brute using **hydra** and **burp**. Start with keeping the Security level to Low and gradually keep on increasing and note your observations. The low and medium are simple. The only change in medium level is that it adds a time delay on failed logins, which only increases the time needed to perform the attack, the attack will succeed anyway. In high security level, the code is using something called a CSRF token, which is like a secret code created by the server and given to the user's device. The hidden token on the webpage needs to be extracted for each login using Burp. See if you can break this security level ☺

To Do: Brute Force Attack on Books Website

Download my Books website, from https://github.com/arifpucit/data-science repository. Just copy all files of this website to /var/www/books/ directory inside your M2 machine. Access it from Kali machine, understand the flow and then launch a brute force attack on it using **hydra** and **burp**.

Disclaimer

The series of handouts distributed with this course are only for educational purposes. Any actions and or activities related to the material contained within this handout is solely your responsibility. The misuse of the information in this handout can result in criminal charges brought against the persons in question. The authors will not be held responsible in the event any criminal charges be brought against any individuals misusing the information in this handout to break the law.