2. Deathnote

Nmap (Network Mapper) :

Purpose: Nmap is a powerful network scanning and reconnaissance tool used to discover hosts, services, and open ports on a network.

Features:
Identifies live hosts and services.
Maps network topology.
Detects vulnerabilities and misconfigurations.
Performs operating system and version detection.

Use Case: Commonly used by security professionals and system administrators for network auditing and penetration testing.

DirBuster :

Purpose: DirBuster is a directory and file brute-forcing tool that discovers hidden directories and files on a web server.

Features:
Uses wordlists to brute-force URLs and directories.
Finds files and directories that are not publicly listed.
Can work with HTTP and HTTPS.

Use Case: Used by penetration testers to find entry points or sensitive data exposed on web servers.

Hydra :

Purpose: Hydra is a password-cracking tool designed to perform brute force attacks on login credentials for various protocols and services.

Features:
Supports numerous protocols (e.g., SSH, FTP, HTTP, RDP).
Allows parallel brute-forcing for faster attacks.
Customizable with wordlists for usernames and passwords.

Use Case: Used by security professionals to test the strength of passwords and identify weak credentials.

Command for ssh connection
ssh username@ip_address

Command for hydra
hydra -L username.txt -P password.txt ssh://victim_ip_address

How to perform privilege escalation
sudo /bin/bash

3. Directory Traversal attack

DirBuster :

Purpose: DirBuster is a directory and file brute-forcing tool that discovers hidden directories and files on a web server.

Features:
Uses wordlists to brute-force URLs and directories.
Finds files and directories that are not publicly listed.
Can work with HTTP and HTTPS.

Use Case: Used by penetration testers to find entry points or sensitive data exposed on web servers.

Directory Traversal Attack :

A Directory Traversal Attack is a security vulnerability that allows attackers to access restricted directories and files outside the intended web application directory. By manipulating file path inputs (e.g., using ../ sequences), attackers can access sensitive files like configuration files, logs, or password files.

Example:
An attacker inputs ../../etc/passwd in a vulnerable URL or form field to access system files.

Mitigation:
Validate and sanitize file path inputs.
Use secure APIs to handle file operations.
Restrict access permissions to sensitive files and directories.

4. Securing the Web application and detecting vulnerabilities

Cross-Site Scripting (XSS) :

Cross-Site Scripting (XSS) is a security vulnerability that occurs when a web application allows malicious scripts to be injected into its web pages. These scripts are then executed in the browser of other users who visit the affected pages. XSS attacks can be used to steal sensitive user data such as cookies, session tokens, or personal information, and they may also allow attackers to manipulate the victim's interaction with the website.

XSS typically arises from insufficient input validation or encoding of user-provided data that is subsequently rendered on the web page. There are four main types of XSS:

Stored XSS: The malicious script is permanently stored on the server (e.g., in a database) and executed whenever a user accesses the infected content.
Reflected XSS: The script is reflected off the server in a response, typically through a crafted URL, and executed when the victim clicks the link.
Form-Based SQL Injection: SQL Injection that exploits input fields in web forms, such as login or search fields, by injecting malicious SQL code.
URL-Based SQL Injection: SQL Injection that manipulates query parameters or input directly in the URL to execute unauthorized SQL commands.

SQL Injection (SQLi):

SQL Injection is a serious web vulnerability where attackers inject malicious SQL code into input fields, manipulating database queries to gain unauthorized access, retrieve sensitive data, bypass authentication, or modify records. It occurs when user input is embedded into queries without proper validation or parameterization.

Types:
Classic SQL Injection: Injecting malicious commands directly into input fields.
Blind SQL Injection: Exploiting responses (true/false) without visible data.
Error-Based SQL Injection: Using error messages to infer database structure.
Union-Based SQL Injection: Combining queries with the UNION operator to extract data.

Mitigation:
Use parameterized queries.
Sanitize and validate user inputs.
Limit database permissions.
Employ security tools like Web Application Firewalls (WAFs).
Proper security practices are essential to prevent SQL Injection and protect databases.

Commands to perform xss :
<script>alert(“vulnerable to xss!”)</script>

Commands to perform sql injection :
' OR '1'='1'#
' AND '1'='1
admin' –
' OR '1'='1
admin'#
' OR 1=1–

5. Broken Authentication and Session Management :

Broken Authentication :
Broken Authentication happens when authentication systems fail, enabling attackers to compromise user accounts. Common causes include weak passwords, reused credentials, session ID exposure, and lack of multi-factor authentication. This can lead to unauthorized access, identity theft, and account takeovers.

Key Points:
Weak password policies and credential reuse are primary causes.
Attackers can exploit session ID vulnerabilities.
Impacts include account hijacking and data theft.
Mitigation: Enforce strong passwords, MFA, and secure session management.

SQL Injection :

SQL Injection is a critical vulnerability where attackers inject malicious SQL code into input fields, manipulating database queries. It can lead to unauthorized data access, database corruption, or bypassing authentication. The vulnerability arises from improper input validation or dynamic SQL queries.

Key Points:
Causes: Improper input sanitization, dynamic queries, excessive privileges.
Impacts: Unauthorized data access, data corruption, full database control.
Mitigation: Use parameterized queries, sanitize inputs, restrict database access.

SQLi cheat sheet :
' OR '1'='1'#
' AND '1'='1
admin' –
' OR '1'='1
admin'#

' OR 1=1–

6. Exploiting XSS in a web application :

Cross-Site Scripting (XSS) :

Cross-Site Scripting (XSS) is a security vulnerability that occurs when a web application allows malicious scripts to be injected into its web pages. These scripts are then executed in the browser of other users who visit the affected pages. XSS attacks can be used to steal sensitive user data such as cookies, session tokens, or personal information, and they may also allow attackers to manipulate the victim's interaction with the website.

XSS typically arises from insufficient input validation or encoding of user-provided data that is subsequently rendered on the web page. There are four main types of XSS:

Stored XSS: The malicious script is permanently stored on the server (e.g., in a database) and executed whenever a user accesses the infected content.
Reflected XSS: The script is reflected off the server in a response, typically through a crafted URL, and executed when the victim clicks the link.
Form-Based SQL Injection: SQL Injection that exploits input fields in web forms, such as login or search fields, by injecting malicious SQL code.
URL-Based SQL Injection: SQL Injection that manipulates query parameters or input directly in the URL to execute unauthorized SQL commands.

Burp Suite :

Burp Suite is a web application security testing tool used by professionals to find vulnerabilities like SQL Injection and XSS. It acts as a proxy to intercept, inspect, and modify web traffic between a browser and a server. Key features include traffic interception, automated vulnerability scanning, and customizable attack payloads.