OWASP TOP10 WEB

A1- INJECTION

- Threats
  - 1- Access, Modify and Delete information in a database
  - 2- Execute Arbitrary system commands on a server
- SQL Injection
  - This occurs when user controlled input is passed to SQL queries. As a result, an attacker can pass in SQL queries to manipulate the outcome of such queries.
- Command Injection
  - This occurs when user input is passed to system commands. As a result, an attacker is able to execute arbitrary system commands on application servers.
  - Blind command injection occurs when the system command made to the server does not return the response to the user in the HTML document. Active command injection will return the response to the user. It can be made visible through several HTML elements.
  - Linux Commands
    - whoami
    - id
    - ifconfig/ip addr
    - uname -a
    - ps -ef
  - Windows Commands
    - whoami
    - ver
    - ipconfig
    - tasklist
    - netstat -an

A2- Broken Authentication

- Threats
  - If an attacker is able to find flaws in an authentication mechanism, they would then successfully gain access to other users' accounts. This would allow the attacker to access sensitive data
- Examples
  - 1- Brute force attacks
  - 2- Use of weak credentials
  - 3- Weak Session Cookies:
- LAB From THM
  - We are given that there is an account named (darren) which contains a flag
    - To access this account, if we try something like(darren) (Notice the space at the end), or even(darren) (3 spaces in the front), for REGISTERING a new account and then we try Logging in with this account. Then we are able to access the account details, in th. is case, the flag from the actual darren account

A3-Sensitive Data Exposure (SDE)

- Explanation
  - When a webapp accidentally divulges sensitive data, we refer to it as "Sensitive Data Exposure". This is often data directly linked to customers (e.g. names, dates-of-birth, financial information, etc), but could also be more technical information, such as usernames and password
  - At more complex levels this often involves techniques such as a "Man in The Middle Attack"
- How to query an SQLite database for sensitive data
  - Threat
    - Well, we can download it and query it on our own machine, with full access to everything in the database. Sensitive Data Exposure indeed!
  - explain
    - In a production environment it is common to see databases set up on dedicated servers, running a database service such as MySQL or MariaDB, however, databases can also be stored as files.
    - These databases are referred to as "flat-file" databases, as they are stored as a single file on the computer. This is much easier than setting up a full database server, and so could potentially be seen in smaller web applications.
  - SQLite3
    - The most common (and simplest) format of flat-file database is an sqlite database. These can be interacted with in most programming languages, and have a dedicated client for querying them on the command line. This client is called " sqlite3", and is installed by default on Kali.
    - exploit
      - 1- download the file
      - 2- To access it we use: sqlite3 <database-name>
      - 3- From here we can see the tables in the database by using the(.tables)command
      - 4- (PRAGMA table_info(customers);)to see the table information
      - (SELECT * FROM customers;) to dump the information from the table

A4- XML External Entity (XXE)

- What is XML?
  - (eXtensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is a markup language used for storing and transporting data.
- Attack Explain
  - An XML External Entity (XXE) attack is a vulnerability that abuses features of XML parsers/data.
  - It often allows an attacker to interact with any backend or external systems that the application itself can access and can allow the attacker to read the file on that system. They can also cause Denial of Service (DoS) attack
  - or could use XXE to perform 'Server-Side Request Forgery (SSRF)' inducing the web application to make requests to other applications. XXE may even enable port scanning and lead to remote code execution
- Types of XXE Attacks
  - 1- An in-band XXE attack is the one in which the attacker can receive an immediate response to the XXE payload.
  - 2- out-of-band XXE attacks (also called blind XXE), there is no immediate response from the web application and attacker has to reflect the output of their XXE payload to some other file or their own server.
- LAB From THM
  - If you just put 2 opening and closing tags, like [<name>M8SZT8</name/>] , then also, the exploit works well
  - Sometime when user generate ssh private and public keys they don't specify a directory where the keys will be stored so the keys get stored in the default directory
    - which is
    - `/home/user/ssh/id_rsa`
    - example
    - `/home/falcon/.ssh/id_rsa`

A5- Broken Access Control

- Explanation
  - Websites have pages that are protected from regular visitors, for example only the site's admin user should be able to access a page to manage other users.
  - If a website visitor is able to access the protected page/pages that they are not authorised to view, the access controls are broken.
- Threats
  - Being able to view "sensitive information"*
  - Accessing unauthorized functionality
- Example Attack Scenario
  - The application uses unverified data in a SQL call that is accessing account ;
  - pstmt.setString(1, request.getParameter("acct")); ResultSet results = pstmt.executeQuery( )
  - An attacker simply modifies the 'acct' parameter in the browser to send whatever account number they want. If not properly verified, the attacker can access any user's account.
    - http://example.com/app/accountInfo?acct=notmyacct
  - An attacker simply force browses to target URLs. Admin rights are required for access to the admin page.
    - http://example.com/app/getappinfo
    - http://example.com/app/admin_getappinfo
  - If an unauthenticated user can access either page, it's a flaw. If a non-admin can access the admin page, this is a flaw
- Insecure Direct Object Reference
  - 
  - is the act of exploiting a misconfiguration in the way user input is handled, to access resources you wouldn't ordinarily be able to access. IDOR is a type of access control vulnerability.
  - THM LAB
    - say we're logging into our bank account, and after correctly authenticating ourselves, we get taken to a URL like this
    - https://example.com/bank?account_number=1234.
    - a hacker may be able to change the account_number parameter to something else like 1235, and if the site is incorrectly configured, then he would have access to someone else's bank information.

A6- Security Misconfiguration

- Explanation
  - It occurs when security could have been configured properly but was not.
  - This vulnerability can often lead to more vulnerabilities, such as default credentials giving you access to sensitive data, XXE or command injection on admin pages.
- Include
  - 1- Poorly configured permissions on cloud services, like S3 buckets
  - 2- Having unnecessary features enabled, like services, pages, accounts or privileges
  - 3- Default accounts with unchanged passwords
  - 4- Error messages that are overly detailed and allow an attacker to find out more about the system
  - 5- Not using HTTP security headers, or revealing too much detail in the Server: HTTP Header

A7- Cross-site Scripting XSS

- Explanation
  - It's a type of injection which can allow an attacker to execute malicious scripts and have it execute on a victim's machine.
  - A web application is vulnerable to XSS if it uses unsanitized user input. XSS is possible in Javascript, VBScript, Flash and CSS
- Main types of cross-site scripting
  - Stored XSS
    - the most dangerous type of XSS. This is where a malicious string originates from the website's database
  - Reflected XSS
    - the malicious payload is part of the victims request to the website
  - DOM-Based XSS - DOM
    - stands for Document Object Model and is a programming interface for HTML and XML documents
    - A web page is a document and this document can be either displayed in the browser window or as the HTML source.
- Scripts to try
  - <script>alert("Got Hacked")</script>
    - Popup's -->> Creates a message popup on a users browser.
  - document.write
    - Writing HTML-->> Override the website's HTML to add your own (essentially defacing the entire page).
  - <http://www.xss-payloads.com/payloads/scripts/simplekeylogger.js.html>
    - XSS Keylogger-->>>>all keystrokes of a user, capturing their password and other sensitive information they type into the webpage.
  - <http://www.xss-payloads.com/payloads/scripts/portscanapi.js.html>
    - Port scanning>>>> A mini local port scanner
  - XSS-Payloads.com (http://www.xss-payloads.com/)
    - Is a website that has XSS related Payloads

A8- Insecure Deserialization

- Explanation
  - Simply, insecure deserialization is replacing data processed by an application with malicious code; allowing anything from DoS to RCE that the attacker can use to gain a foothold in a pentesting scenario.
  - Specifically, this malicious code leverages the legitimate serialization and deserialization process used by web applications.
  - "Insecure Deserialization is a vulnerability which occurs when untrusted data is used to abuse the logic of an application"
- What's Vulnerable?
  - Any application that stores or fetches data where there are no validations or integrity checks in place for the data queried or retained
  - A few examples of applications of this nature are:
    - E-Commerce Sites
    - Forums
    - API's
    - Application Runtimes (Tomcat, Jenkins, Jboss, etc)
- De(Serialization)
  - What does this mean?
    - Serialisation is the process of converting objects used in programming into simpler, compatible formatting for transmitting between systems or networks for further processing or storage.
    - Serialisation
    - deserialisation is the reverse of this; converting serialised information into their complex form - an object that the application will understand.
    - Deserialisation
    - Say you have a password of "password123" from a program that needs to be stored in a database on another system. To travel across a network this string/output needs to be converted to binary.
    - Of course, the password needs to be stored as "password123" and not its binary notation. Once this reaches the database, it is converted or deserialised back into " password123" so it can be stored.
- Cookies
  - 
  - Cookies are an essential tool for modern websites to function. Tiny pieces of data, these are created by a website and stored on the user's computer.
  - you'll notice cookies store login information like the Above!
- THM LAB
  - 1- Creating a new cookie field,
  - 2- Opening a form,
  - 3- Making a python script to create a Base64 Encoded Cookie,
  - 4- Opening a netcat listener,
  - 5- Changing the cookie value in the new field,
  - 6- And finally, getting a reverse shell to the Website's Server.
  - After getting a reverse shell, a simple cd . and an ls would do.

A9- Components With Known Vulnerabilities

- Explanation
  - If a company misses a single update for a program they use, they could be vulnerable to any number of attacks.
  - For example, let's say that a company hasn't updated their version of WordPress for a few years, and using a tool such as wpscan,
  - you find that it's version 4.6. Some quick research will reveal that WordPress 4.6 is vulnerable to an unauthenticated remote code execution(RCE) exploit, and even better you can find an exploit already made on exploit-db.
- THM LAB
  - Search Online Book Store 1.0 in https://www.exploit-db.com/
  - Download the exploit and set the target and run script
  - How many characters are in </etc/passwd> use <<<wc -c /etc/passwd >>>

A10-Insufficient Logging and Monitoring

- suspicious activity includes
  - 1- multiple unauthorised attempts for a particular action
  - 2- requests from anomalous IP addresses or locations
  - 3- use of automated tools
  - 4- common payloads
- Threats
  - 1- regulatory damage: if an attacker has gained access to personally identifiable user information and there is no record of this, not only are users of the application affected, but the application owners may be subject to fines or more severe actions depending on regulations.
  - 2- risk of further attacks: without logging, the presence of an attacker may be undetected. This could allow an attacker to launch further attacks against web application owners by stealing credentials, attacking infrastructure and more.
- Explanation
  - When web applications are set up, every action performed by the user should be logged. Logging is important
  - because in the event of an incident, the attackers actions can be traced. Once their actions are traced, their risk and impact can be determined