WEB APPLICATION SECURITY

LABORATORY QUESTIONS

1. Install wireshark and explore the various protocols:
   (a) Analyze the difference between HTTP vs HTTPS.
   (b) Analyze the various security mechanisms embedded with different protocols.

2. Identification of the vulnerabilities using OWASP ZAP tool.

3. Create simple REST API using Python & Perform HTTP Requests.

4. Installation of Burp Suite to Perform Vulnerabilities Testing.

5. Attacking a Website Using Social Engineering Method.

6. Use Wireshark to capture and analyze SSL/TLS traffic, understanding how these protocols secure data in transit.

7. Analyze the session management mechanisms of a web application using browser developer tools to understand how sessions are created and managed.

8. Explore various security testing tools provided by OWASP (e.g., OWASP ZAP, OWASP Dependency-Check) to understand their functionalities and usage.

9. Use browser developer tools or a tool like Wireshark to analyze HTTP headers exchanged between a client and server, focusing on security-related headers like Content-Security-Policy and X-Frame-Options.

10. Analyze the cookies used by a web application to understand how session management and authentication are handled, and identify any potential security risks.

11. Use a tool like Burp Suite to test a website's protection against clickjacking by attempting to overlay content from another site onto the target site.

12. Develop a basic security incident response plan for a web application, outlining steps to take in case of a security breach.

13. Cross-Origin Resource Sharing (CORS) Analysis
   (a) Use a web browser's developer tools to inspect CORS headers in HTTP responses.
   (b) Visit a website that makes cross-origin requests and analyze the CORS headers to understand the cross-origin policy implemented by the server.

14. Content Security Policy (CSP) Evaluation. Use a web security testing tool (e.g., CSP Evaluator) to analyze the CSP implementation of a website.

15. HTTP Public Key Pinning (HPKP) Assessment. Use a web security testing tool (e.g., HPKP Analyzer) to analyze the HPKP implementation of a website.

06. Use Wireshark to capture and analyze SSL/TLS traffic, understanding how these protocols secure data in transit.

1. Download and install Wireshark from the official website (https://www.wireshark.org/). Install WinPcap or Npcap when prompted during the installation process.

2. Launch Wireshark. You might need administrative privileges to capture network traffic.

3. Start Capturing Traffic:
- Select the network interface you want to capture traffic from.
- Click on the interface and then click the "Start" button to begin capturing traffic.

4. To focus on SSL/TLS traffic, apply display filter. Type `ssl` in the display filter toolbar and press Enter. This will show only SSL/TLS traffic.

5. Analyze the SSL/TLS Traffic:
- Look for packets with the SSL/TLS protocol (indicated by the "Protocol" column in Wireshark).
- Right-click on a packet, select "Follow", then "SSL Stream" to view the decrypted content of the SSL/TLS session, if the traffic is not encrypted (e.g., if you have access to the private key).

6. Inspect SSL/TLS Handshake:
- Look for packets containing the SSL/TLS handshake, which establishes the secure connection.
- Analyze the handshake process, including the ClientHello, ServerHello, Certificate, and Finished messages, to understand how the SSL/TLS protocol negotiates a secure connection.

7. Stop Capturing. When you're finished capturing traffic, click the "Stop" button in Wireshark.

8. To save the captured SSL/TLS traffic for further analysis, go to "File" > "Export Objects" > "HTTP" and save the packets as a .pcap file.

07. Analyze the session management mechanisms of a web application using browser developer tools to understand how sessions are created and managed.

1. Open your web browser and navigate to the web application you want to analyze. Right-click on the page and select "Inspect" or press `Ctrl+Shift+I` (Cmd+Option+I on Mac) to open the developer tools.

2. In the developer tools, find and click on the "Network" tab. This tab allows you to monitor network activity, including HTTP requests and responses.

3. If there is existing network data, click the "Clear" button or press `Ctrl+R` to clear it. This will ensure you're capturing fresh data.

4. Capture Network Traffic:
- Perform actions on the web application that trigger session management, such as logging in, accessing restricted content, or performing transactions.
- As you interact with the application, the Network tab will capture the HTTP requests and responses.

5. Analyze Requests and Responses:
- Look for requests that contain session-related information, such as cookies or headers.
- Look for cookies named typically with `session`, `SID`, or similar, which store session identifiers. The value of these cookies uniquely identifies your session.
- Check for headers like `Set-Cookie` in responses and `Cookie` in subsequent requests, which contain session information.
- Some applications may use URL parameters or hidden form fields to manage sessions. Look for these patterns in the request and response data.

6. Inspect Session Cookies:
- Click on a request that contains a session cookie.
- In the Headers tab of the request details, find the "Cookies" section to view the session cookie(s) sent with the request.

7. View Session Details:
- Look for requests that access protected resources or perform actions that require authentication.
- Inspect the request headers or cookies to see how the session is being managed.
- Look for the expiration time of the session cookie to understand the session's duration.

8. Observe Session Expiry and Renewal:
- Log out of the application and observe how the session is invalidated or expired.
- Log back in to see how a new session is created or how the session is renewed.

08. Explore various security testing tools provided by OWASP (e.g., OWASP ZAP, OWASP Dependency-Check) to understand their functionalities and usage.

OWASP ZAP (Zed Attack Proxy):

- Functionality:
    - OWASP ZAP is a security testing tool designed to find vulnerabilities in web applications during development and testing. It can be used to intercept and modify HTTP/HTTPS requests and responses, perform automated scans for common security issues, and generate reports.
- Steps:
    - Start ZAP and configure your browser to use ZAP as a proxy.
    - Explore your web application, and ZAP will intercept and display the requests and responses.
    - Use the automated scanner to scan for vulnerabilities like XSS, SQLi, CSRF, etc.

OWASP Dependency-Check:

- Functionality:
    - OWASP Dependency-Check is a tool used to identify known vulnerabilities in project dependencies. It scans dependencies (e.g., libraries, components) used in a project against a database of known vulnerabilities (e.g., NVD, Retire.js) and generates a report.

- Steps:
    - Configure Dependency-Check for your project, specifying the type of dependencies to scan (e.g., Java libraries, Node.js modules).
    - Run the scan to identify vulnerabilities in the dependencies.
    - Review the report to see the list of vulnerabilities and take appropriate actions (e.g., update dependencies, apply patches).

OWASP Amass:

- Functionality:
    - OWASP Amass is a tool used for network mapping and attack surface discovery. It helps security professionals to gather information about the target network, such as subdomains, IP addresses, and associated domains.

- Steps:
    - Provide a domain name or IP address range to scan.
    - Amass will perform various techniques (e.g., DNS enumeration, web scraping) to gather information.
    - Review the output to identify potential attack surfaces and security risks.

OWASP DefectDojo:

- Functionality:
    - OWASP DefectDojo is a vulnerability management tool used to track and manage security vulnerabilities in applications. It helps teams to prioritize and remediate vulnerabilities efficiently.

- Steps:
    - Import vulnerabilities from various sources (e.g., ZAP, Dependency-Check, manual testing).
    - Assign vulnerabilities to team members for remediation.
    - Track the progress of remediation efforts and generate reports.

09. Use browser developer tools or a tool like Wireshark to analyze HTTP headers exchanged between a client and server, focusing on security-related headers like Content-Security-Policy and X-Frame-Options.

Using Browser Developer Tools:

1. Open your web browser and navigate to the website you want to analyze. Right-click on the page and select "Inspect" or press `Ctrl+Shift+I` (Cmd+Option+I on Mac) to open the developer tools.

2. In the developer tools, find and click on the "Network" tab. This tab allows you to monitor network activity, including HTTP requests and responses.

3. Capture Network Traffic:
   - Perform actions on the web application that trigger HTTP requests and responses.
   - As you interact with the application, the Network tab will capture the HTTP headers exchanged between the client and server.

4. Inspect HTTP Headers:
   - Click on a request in the Network tab to view its details.
   - In the Headers tab of the request details, look for security-related headers like `Content-Security-Policy` and `X-Frame-Options`.
   - Analyze the values of these headers to understand the security policies implemented by the server.

Using Wireshark:

1. Launch Wireshark and start capturing traffic on the network interface used by your web browser.

2. Capture HTTP Traffic:
   - Perform actions on the web application that trigger HTTP requests and responses.
   - Wireshark will capture the raw network packets exchanged between the client and server.

3. Apply a Filter:
   - To focus on HTTP traffic, you can apply a display filter by typing ‘http’ in the filter toolbar and pressing Enter.
   - To filter for specific headers, you can use a filter like ‘http.request.headers’ or ‘http.response.headers’.

4. Analyze HTTP Headers:
   - Look for packets that contain the HTTP headers you're interested in, such as ‘Content-Security-Policy’ and ‘X-Frame-Options’.
   - Right-click on a packet, select "Follow", then "TCP Stream" to see the entire HTTP conversation and the headers exchanged.

5. Inspect Security Headers:

- Analyze the values of the security-related headers to understand the security policies implemented by the server.
- Pay attention to directives like ‘frame-ancestors’ in ‘X-Frame-Options’ and directives in ‘Content-Security-Policy’ to see how they restrict certain behaviors.

10. Analyze the cookies used by a web application to understand how session management and authentication are handled, and identify any potential security risks.

1. Open your web browser and navigate to the web application you want to analyze. Right-click on the page and select "Inspect" or press ‘Ctrl+Shift+I’ (Cmd+Option+I on Mac) to open the developer tools.

2. In the developer tools, find and click on the "Application" tab. This tab allows you to inspect cookies, local storage, and other web storage mechanisms.

3. Locate Cookies:
- In the "Application" tab, expand the "Cookies" section in the sidebar.
- Look for cookies related to the web application, which are usually named with ‘session’, ‘SID’, or similar, and may include authentication-related information.

4. Analyze Cookie Properties:
- Click on a cookie to view its properties, including its name, value, domain, path, expiration date, and flags (e.g., Secure, HttpOnly).
- Pay attention to the expiration date, as cookies that persist for a long time may pose a security risk if they are not properly invalidated or if they contain sensitive information.

5. Inspect Authentication Cookies:
- Look for cookies that are used for authentication, such as those that contain a session identifier (‘SID’) or a token.
- Verify if these cookies are encrypted or if they are sent over HTTPS to prevent interception.

6. Identify Security Risks:
- Check if cookies are marked as ‘HttpOnly’ to prevent client-side scripts from accessing them, which helps mitigate against cross-site scripting (XSS) attacks.
- Look for the ‘Secure’ flag, which ensures that cookies are only sent over HTTPS, preventing them from being transmitted over unencrypted connections.
- Ensure that cookies related to sensitive operations, such as authentication, have a short expiration time to limit their lifespan and reduce the risk of session hijacking.

7. Check for Secure and HttpOnly Flags:
- Look for cookies that are missing the ‘Secure’ flag, which could allow them to be transmitted over unencrypted connections.
- Check if cookies are missing the ‘HttpOnly’ flag, which could allow client-side scripts to access them, making them vulnerable to XSS attacks.

11. Use a tool like Burp Suite to test a website's protection against clickjacking by attempting to overlay content from another site onto the target site.

1. Ensure Burp Suite is properly configured and running. Set up your browser to proxy through Burp.

2. Use your browser to navigate to the website you want to test.

3. In Burp, go to the "Proxy" tab and ensure the "Intercept is on" button is pressed.

4. Refresh the page on the target website. Burp should capture the request in the "Proxy" tab.

5. Right-click on the captured request in Burp and select "Send to Repeater." In the Repeater tab, you can modify the request to include a custom header to test for clickjacking protection. For example, you can add a "X-Frame-Options" header with a value of "deny" or "sameorigin".

6. After modifying the request, click "Go" to send it.

7. Check the response in the Repeater tab to see if the website's protection against clickjacking is effective. If the website is properly protected, it should prevent the content from being displayed in an iframe.

12. Develop a basic security incident response plan for a web application, outlining steps to take in case of a security breach.

1. Preparation:

- Designate a team responsible for responding to security incidents. This team should include individuals from IT, security, legal, and management.
- Establish criteria for categorizing the severity of incidents to prioritize response efforts.
- Create a communication plan for notifying stakeholders, including internal teams, customers, and regulatory bodies, if necessary.
- Set up systems to monitor the web application for suspicious activity and maintain logs for analysis.

2. Detection and Analysis:

- Use monitoring tools to detect unusual behavior or unauthorized access.
- Assess the impact of the security incident on the web application, data, and users.
- Take immediate steps to contain the incident to prevent further damage.

3. Response:

- Implement measures to mitigate the impact of the security breach, such as patching vulnerabilities or blocking malicious activity.
- Conduct forensic analysis to understand the root cause of the incident and gather evidence for legal or regulatory purposes.
- Notify relevant stakeholders, including customers, partners, and regulatory bodies, as per the communication plan.
- Ensure compliance with applicable laws and regulations, such as GDPR or CCPA, regarding data breach notifications and reporting.

4. Recovery:

- Restore the web application to normal operation after ensuring that the security vulnerabilities have been addressed.
- Conduct a post-incident review to identify lessons learned and improve incident response procedures for future incidents.
- Update the incident response plan based on the lessons learned from the security incident.

5. Training and Awareness:

- Provide regular training to the incident response team and relevant stakeholders on security best practices and incident response procedures.

- Raise awareness among employees and users about security threats and how to report suspicious activity.

6. Documentation and Reporting:

- Keep detailed records of the incident, including timelines, actions taken, and outcomes.
- Report the incident to relevant authorities, such as law enforcement or regulatory bodies, as required by law.

7. Continuous Improvement:

- Regularly review and update the incident response plan based on new threats, vulnerabilities, and lessons learned from previous incidents.
- Conduct regular exercises, such as tabletop simulations, to test the effectiveness of the incident response plan and improve response capabilities.

13. Cross-Origin Resource Sharing (CORS) Analysis:

(a) Use a web browser's developer tools to inspect CORS headers in HTTP responses.

(b) Visit a website that makes cross-origin requests and analyze the CORS headers to understand the cross-origin policy implemented by the server.

(a) Use a web browser's developer tools to inspect CORS headers in HTTP responses.

1. Open your web browser (e.g., Chrome, Firefox, Edge) and navigate to the website you want to analyze.

2. Press `F12` or right-click on the page and select "Inspect" to open the developer tools.

3. In the developer tools, navigate to the "Network" tab.

4. Reload the page to capture the network requests and responses.

5. Locate the request in the network log that you want to analyze. This could be a request made by JavaScript code to a different origin (cross-origin request).

6. Click on the request in the network log to view its details. Look for the "Response Headers" section.

7. Look for the following CORS-related headers in the response headers:

- '**Access-Control-Allow-Origin**': Specifies which origins are allowed to access the resource. If this header is present and its value is "*", any origin is allowed. If it specifies a specific origin, only that origin is allowed.
- '**Access-Control-Allow-Methods**': Specifies the HTTP methods (e.g., GET, POST) allowed when accessing the resource.
- '**Access-Control-Allow-Headers**': Specifies which headers are allowed in the actual request.

8. Based on the values of these headers, you can determine the cross-origin policy implemented by the server.
For example, if 'Access-Control-Allow-Origin' is "*", the server allows requests from any origin. If it specifies a specific origin, the server restricts requests to that origin.

(b) Visit a website that makes cross-origin requests and analyze the CORS headers to understand the cross-origin policy implemented by the server.

1. Go to a website that makes cross-origin requests. For example, you could visit a site that uses AJAX to fetch data from a different domain.

2. Open the developer tools in your web browser and go to the "Network" tab. Make sure the "Preserve log" option is enabled to keep track of network activity.

3. Interact with the website to trigger cross-origin requests. For example, if the site loads data from a different domain when you click a button or scroll down, perform that action.

4. Look for the network requests in the developer tools that are labeled as cross-origin requests. These requests will have an "Origin" header indicating the requesting domain and may have CORS-related headers in the response.

5. For each cross-origin request, analyze the CORS-related headers in the response, including `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods`, and `Access-Control-Allow-Headers`.

6. Based on the CORS headers, determine the cross-origin policy implemented by the server. Pay attention to whether the server allows requests from any origin (`*`) or only from specific origins.

14. Content Security Policy (CSP) Evaluation. Use a web security testing tool (e.g., CSP Evaluator) to analyze the CSP implementation of a website.

1. Go to the CSP Evaluator website at https://csp-evaluator.withgoogle.com/

2. Enter the URL of the website you want to analyze in the text field provided on the CSP Evaluator homepage.

3. Click on the "Start Evaluation" button to initiate the CSP evaluation process.

4. After the evaluation is complete, CSP Evaluator will display a report detailing the website's CSP implementation. This report will include information about the effectiveness of the CSP in protecting against various types of attacks, such as XSS (Cross-Site Scripting).

5. Review the evaluation results to understand how well the website's CSP is configured. Pay attention to any recommendations or warnings provided by CSP Evaluator regarding potential vulnerabilities or areas for improvement in the CSP implementation.

6. If CSP Evaluator identifies any issues or weaknesses in the CSP implementation, take steps to address them. This may involve modifying the website's CSP directives to strengthen security.

7. After making changes to the CSP implementation, re-evaluate the website using CSP Evaluator to ensure that the improvements have been effective and that the CSP provides adequate protection against security threats.

15. HTTP Public Key Pinning (HPKP) Assessment. Use a web security testing tool (e.g., HPKP Analyzer) to analyze the HPKP implementation of a website.

1. Go to the HPKP Analyzer tool at https://report-uri.com/home/pkp_analyse

2. Enter the URL of the website you want to analyze in the text field provided on the HPKP Analyzer page.

3. Click on the "Analyze" button to initiate the HPKP analysis process.

4. After the analysis is complete, the HPKP Analyzer will display a report detailing the HPKP implementation of the website. This report will include information about the pinned public keys, expiration dates, and any issues or warnings detected.

5. Review the analysis results to understand how well the website's HPKP is configured. Pay attention to any recommendations or warnings provided by the HPKP Analyzer regarding potential issues or areas for improvement in the HPKP implementation.

(Alternative)

Use a web proxy tool like Burp Suite to test the access control mechanisms of a web application, attempting to access restricted resources or perform actions that should be limited to specific users or roles.

1. Setup Burp Suite:
   - Configure your browser to proxy through Burp Suite.
   - Open Burp Suite and ensure interception is on.

2. Use your browser to navigate to the web application you want to test.

3. Perform actions in the web application that should be restricted to certain users or roles. Use Burp Suite's Intercept feature to capture the requests.

4. Once a request is intercepted, right-click on it and select "Send to Repeater."

5. In the Repeater tab, modify the request to attempt to access a restricted resource or perform a restricted action.
For example, if the application uses URL parameters or cookies for access control, modify these values to test different scenarios.

6. After modifying the request, click "Go" to forward it to the server.

7. Examine the response from the server to see if the access control mechanisms are working as expected. Look for error messages, access denied responses, or other indicators that the action was not allowed.

8. Repeat the process for different scenarios to thoroughly test the access control mechanisms.