|  |  |
| --- | --- |
| Internship Project Title | RIO-125: Applying Dynamic Application Security Testing Tools to Find Defects in Web Applications |
| Project Title | Applying Dynamic Application Security Testing Tools to Find Defects in Web Applications |
| Name of the Company | TCS iON |
| Name of the Industry Mentor | JAGRUTI DEORE |
| Name of the Institute | K.B.N College |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Start Date | End Date | Total Effort (hrs.) | Project Environment | Tools used |
| 16-07-2022 | 18-08-2022 | 125 | Windows | ZAP, Burp suite |
| * A | | | | |
| Solution Approach:  **Difference Between SAST and DAST:**   * **DAST** * Black box testing (does not require source code). * This testing is performed at end of SDLC. * Discovers run-time and environment related issues. * More expensive. * This type testing is a hackers approach of testing. * Supports less variety of software. * **SAST** * White box testing (Requires source code). * This testing is performed at beginning of SDLC. * Cannot discover run-time and environment related issues. * Less expensive. * This type testing is a developers approach of testing. * Supports all sorts of code/software.   **OWASP TOP 10 VULNERABILITIES:**   1. **Broken Access Control:**   Bypassing access control checks by modifying the URL or by using an attack tool modifying API requests.   1. **Cryptographic Failures:**   The organization needs to check proper security protocols to protect the user’s personal information.   1. **Injection:**   Injection is a type of attack allows an attacker to inject code into a program or query onto a computer in order to execute remote commands that can read or modify a database, or change data on a web site.   1. **Insecure Design:**   Insecure design is the lack of security controls. If the application is not designed with proper security, hackers may use this flaw.   1. **Security Misconfiguration:**   Security misconfigurations are security controls that are left insecure, putting your systems and data at risk.   1. **Vulnerable and Outdated Components:**   Component-based vulnerabilities occur when a software component is unsupported, out of date, or vulnerable to a known exploit.   1. **Identification and Authentication failures:**   Identification and authentication failures are vulnerabilities related to applications’ authentication schemes. Such failures can lead to serious and damaging data breaches.   1. **Software and Integrity Failures:**   Software and data integrity failures frequently occur when the code implementation and the underlying infrastructure lack the ability to protect the code against all integrity violations.   1. **Security Logging and Monitoring Failures:**   Every time the user accesses the web application, The web application needs to monitor every move of the user.   1. **Server-Side Request Forgery:**   A Server-Side Request Forgery (SSRF) attack involves an attacker accessing server functionality to access or modify resources. The attacker targets an application that supports data imports from URLs or allows them to read data from URLs.  **Phase 1:**   1. **SQL Injection:**  * SQL injection is a code injection technique that might destroy your database. * SQL injection is one of the most common web hacking techniques. * SQL injection is the placement of malicious code in SQL statements, via web page input.  1. **Location:-** [**http://localhost:3000/#/login**](http://localhost:3000/#/login) 2. **Finding vulnerability using Burp Suite:**   This is the website we are finding vulnerability.    Here, I am using the Burp Suite to find SQL injection vulnerabilities in the website.  **Burp Suite:**  Burp Suite is a set of tools used for penetration testing of web applications.    **BEFORE ATTACK:-**  **Step 1:**   * Open website and Burp suite.      * Burp Suite.     **Step 2:**   * Openthe login page and enter a random email id and password.     **Step 3:**   * Open Burp Suite and click on Intercept.     **Step 4:**   * Now click on “login” on the website.     **Step 5:**   * Open the Burp Suite and click on proxy. * We see a request for logging in with admin.     **Step 6:**   * Right click on the page and click on “Send to Repeater”.     **Step 7:**   * Now click on “send” to send the request again to the server.      * Still we are getting the Invalid email or password.     **Step 8:**  Here, we are using the SQL Query (“admin’ or 1=1 –") to tell the database that the rest of the query is comment.     * We get an authentication token which is JSON token.     **Step 9:**   * Copy the token and decode on the website.   (<https://jwt.io/>)     * By decoding it, we will get the payload list, which contains email and password in hash form.     **Step 10:**   * Decode the password in this website.   (<https://hashes.com/en/decrypt/hash>)   * By decoding it, we got the password as “admin123”.     **Step 11:**   * Now we are using the credentials to login to the website.   **Email:** ([admin@juice-sh.op](mailto:admin@juice-sh.op)) and **Password:** (admin123).    **AFTER ATTACK:-**   * We have successfully logged into the website using admin’s credentials.      * We can see the admin’s profile.      1. **Risk/ Undesirable impact if exploited:**  * A successful attack can gain access to all the internal information within the database, such as emails and passwords. * The attacker has the ability to change the data.  1. **How to fix the defect:**  * It requires developers to build SQL statements that are parametrized. * Identify the essential SQL statements and establish a whitelist for all valid SQL statements, leaving unvalidated statements out of the query. This process is known as input validation or query redesign.  1. **Authoritative reference links, and explanatory examples:**  * Reference Link: <https://www.youtube.com/watch?v=3Axp3VDnf0I> * Reference Link: <https://www.w3schools.com/sql/sql_injection.asp>  1. **Cross Site Scripting (XSS):**  * Cross Site Scripting (XSS) is a code injection attack executed on the client-side of a web application. * There are three types of cross site scripting attacks.  1. *Reflected:* In this case, the data is not stored on the web server. 2. *DOM:* DOM which makes use of the document object to inject the malicious script. 3. *Stored:* In this case, the malicious script is stored on the web server. 4. **Location:-** [**http://localhost:3000/#/login**](http://localhost:3000/#/login) 5. **Here we are doing a reflected XSS attack on this website:**   In this case, the data is not stored on the web server.  **BEFORE ATTACK:-**   * This is the website we are finding vulnerability.     **Step 1:**   * Open the website and create an account. After that, log into the account.     **Step 2:**   * Add any fruit to the basket.     **Step 3:**   * Click on the “Your Basket” section to see the fruits that we have added to the cart.      * Fruits that we have added to the cart.     **Step 4:**   * Now, click on checkout.     **Step 5:**   * Click on “Add New Address” and fill in any address then click on “Continue”.     **Step 6:**   * Choose a delivery speed option among three options and then click on “Continue”.       **Step 7:**   * Choose any payment method and then click on “Continue”.     **Step 8:**   * Click on “Place your order and pay”.     **Step 9:**   * Now, click on “Track orders”.     **Step 10:**   * URL: <http://localhost:3000/#/track-result?id=5267-78ebe55631f33dca> * Here, we can see a parameter in the URL.     **Step 11:**   * Now we are using this parameter to show a popup whenever a user clicks on this URL. * Here, we are adding this script(**<iframe src="javascript:alert(`xss`)">**) in place of this parameter.   (An iFrame injection XSS is a common cross-site scripting attack that combines malicious JavaScript with an iframe that loads a legitimate page in an effort to steal data from an unsuspecting user.)   * After that reload the page.     **AFTER ATTACK:-**   * We get a popup message saying that the page has been hacked.      1. **Risk/ Undesirable impact if exploited:-**   Cross-site scripting vulnerabilities normally allow an attacker to play as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data.   1. **How to fix the defect:-**  * The best way to prevent Cross-Site Scripting attacks is to limit allowable user inputs. We can do this by establishing a list of restricted characters called the blacklist. * For example, we can prevent a user from inputting certain script characters like greater than or less than symbols or quotes. You could also compare inputs against the lists of untrusted character phrases, such as script tags, that are known to be dangerous or malicious.  1. **Authoritative reference links, and explanatory examples:**  * Reference Link: <https://www.youtube.com/watch?v=cWu_FJUrH5Y&t=776s> * Reference Link:   <https://www.youtube.com/watch?v=PPzn4K2ZjfY>   1. **Broken Authentication:-**   Broken authentication means when an attacker tries to bypass the login credentials of a victim user using different tools and gaining access to the victim user account.   1. **Location:-** [**http://localhost:3000/**](http://localhost:3000/) 2. **Bruteforce the Administrator account’s password:**   This is the website we are finding vulnerability.    Here we will be using Burp Suite to perform a Bruteforce attack using an intruder and bypass login.    **BEFORE ATTACK:-**  **Step 1:**   * Open website and click on any product.     **Step 2:**   * Click on the “Reviews”.      * Here, we found an email ID (admin@juice-sh.op). Now, we bruteforce this email ID using different passwords and try to login to the admin’s account.     **Step 3:**   * Now go to the login page and enter the email ID we found in the "Reviews" section and enter some random password.     **Step 4:**   * Open the Burp Suite and turn on the Burp Suite intercept.     **Step 5:**   * Click on the Login.     **Step 5:**   * As soon as we clicked on login, we found a login request in the Burp Suite.     **Step 6:**   * Right click on the burp suite and click on “Send to intruder”.     **Step 7:**   * We have already sent the request to the intruder. So, turn off the intercept.     **Step 8:**   * Click on the intruder.     **Step 9:**   * Now, we do not want to make all these parts dynamic, we only want the password part to be dynamic. So, click on clear.     **Step 10:**   * We only want to make the password part dynamic, so select the password and click on add. so that we can send different passwords to the selected part only.     Select the password  **Step 11:**   * Now, click on payloads.     **Step 12:**   * Now, we are going to paste some of the sample passwords here. These passwords will be sent in place of the selected password.     **Step 13:**   * Click on the “start attack”.     **Step 14:**   * Here, we found a difference in length.     **Step 15:**   * Click on the “Response”. * Here, we can see a token. So, the password is “admin123”. * Stop the attack.     Token  **Step 16:**   * Now we are using the credentials to login to the website.   **Email:** ([admin@juice-sh.op](mailto:admin@juice-sh.op)) and **Password:** (admin123).    **AFTER ATTACK:-**   * We have successfully logged into the admin’s account using the credentials.      * We can see the admin’s profile.      1. **Risk/ Undesirable impact if exploited:-**   A successful attack can result in a malicious attacker gaining complete access to all the data in the web application. The attacker can access the data, manipulate the data, and expose the data.   1. **How to fix the defect:-**  * Enable multi-factor authentication. * Implement strong password policies. * Use a web application firewall. * Limited failed login attempts. * Secure session access.  1. **Authoritative reference links, and explanatory examples:-**   Reference Link: <https://www.youtube.com/watch?v=mruO75ONWy8>  Reference Link: https://curiositykillscolby.com/2020/11/15/pwning-owasps-juice-shop-pt-19-password-strength/   1. **Broken Access Control:-**   Broken Access Control issues occur if a malicious user tries to access higher privileged functionalities by changing the URL endpoint or reference values which refers to the functionality.   1. **Location:-** [**http://localhost:3000/#/**](http://localhost:3000/#/) 2. **Accessing the cart of another person in a shopping web application:**   This is the website we are finding vulnerability.    **BEFORE ATTACK:-**  **Step 1:**   * Open the website click on account and click on login.     **Step 2:**   * Login with your registered email ID or create an account and then login.     Create a account  **Step 3:**   * After logging to your account, add any fruits to the basket.     **Step 4:**   * Click on “your basket”.     **Step 5:**   * Open burp suite and click on “proxy”.     **Step 6:**   * Now, Click on “Http history”. * Here, we can see some requests.     **Step 7:**   * Now, select this particular request (/rest/basket/8).     **Step 8:**   * Now, right click on the interface and this request to “Repeater”.     **Step 9:**   * Click on “Repeater”. * We can see that this is a particular request.     **Step 10:**   * Once if we send a request, we can see the items in our cart here. So, click on send.     Items in our cart  **Step 11:**   * (/rest/basket/8) is the URL and “8” is related to my account. * If we change the value “8” to another value, we can see another user’s cart.     **Step 12:**   * Here we are changing the value to “1” in place of “8”.     **Step 13:**   * After changing the value click on “send”.     **AFTER ATTACK:-**   * Here we can see the cart values of another user, which contain an id of "1".      1. **Risk/ Undesirable impact if exploited:-**   Broken access controls can put applications at risk of a data breach, usually resulting in the loss of data. A successful attack can steal information accessed by users of the application, manipulate data by performing actions.   1. **How to fix the defect:-**  * Handle access control at server-side. * Constant testing and auditing of access controls. * Clean code with binary access controls. * Proper session management.  1. **Authoritative reference links, and explanatory examples:-**   Reference Link: <https://www.youtube.com/watch?v=K2PiHznrPB>  Reference Link: <https://portswigger.net/web-security/access-control>  **5. Improper input validation:**  If any software does not validate input, it will affect the control flow. When software does not validate input properly, an attacker is able to manipulate the input in a form that is not expected by the rest of the application. This may be used by the attacker for security exploits.   1. **Location:-** <http://localhost:3000/#/contact> 2. **Giving a zero-star rating to an online store:**   **BEFORE ATTACK:-**  **Step 1:**   * Open the website.     **Step 2:**   * Click on the side menu on the left side.     **Step 3:**   * Click on “customer feedback”.     **Step 4:**   1. Enter a random comment. 2. Give a rating. (It is not accepted that a rating begins with zero). 3. Enter the captcha.     **Step 5:**   * We are going to submit a zero star, which is not accepted by the website. * Open the Burp Suite 🡪 Click on Proxy.     **Step 6:**   * Now, Click on “intercept on”.     **Step 7:**   * Open the website and click on “Submit”.     **Step 8:**   * We see a request here.     **Step 9:**   * Change the rating to “0”.     **Step 10:**   * Click on “Forward” to forward the request.     **AFTER ATTACK:-**   * We have successfully submitted the feedback with a zero star, which is not accepted by the website.      1. **Risk/ Undesirable impact if exploited:-**   Improper validation of form data is one of the main causes of security vulnerabilities. It exposes websites to attacks such as header injections, cross-site scripting, and SQL injections.   * Header injection attacks can be used to send email spam from your web server. * cross-site scripting may allow an attacker to post any data to your site. * SQL injection may corrupt your database backend.  1. **How to fix the defect:-**  * Better input validation. * Safe programming techniques. Techniques for detecting potential buffer overflows in code.  1. **Authoritative reference links, and explanatory examples:-**   Reference Link: https://www.youtube.com/watch?v=Xpi0ePeviX8  Reference Link: <https://cwe.mitre.org/data/definitions/20.html#:~:text=When%20software%20does%20not%20validate,resource%2C%20or%20arbitrary%20code%20execution>.   1. **Sensitive Data Exposure:-**   Sensitive Data Exposure occurs when an organization unknowingly exposes sensitive data or when a security incident leads to the accidental or unlawful destruction, loss, alteration, or unauthorized disclosure of, or access to sensitive data.   1. **Location:** <http://localhost:3000/#/about> 2. **Exposing confidential documents on a website:**   **BEFORE ATTACK:-**  **Step 1:**   * Open the website and click on side menu bar.     **Step 2:**   * Click on “About Us”.     **Step 3:**   * We have found this interesting link.     **Step 4:**   * Open Burp Suite and click on Proxy -> HttpHistory. * We can see some requests here.     **Step 5:**   * Now, we are going to use this particular request (/ftp/legal.md).     **Step 6:**   * Send this request to repeater. Right click -> “Send to Repeater”.     **Step 7:**   * Click on “Repeater”.     **Step 8:**   * Delete (legal.md) in the request.   **Before:**    **After:**    **Step 9:**   * Click on “Send” to resend the request”.     **Step 10:**   * We can see some responses here. * Scroll down in the responses.     **Step 11:**   * We can see the directory of the files, and we found an interesting file (acquisitions.md).     **Step 12:**   * Paste the “acquisitions.md” here and click on “send” to send the request.     Paste  **After attack:-**   * We can see that this document is “confidential”.      1. **Risk/ Undesirable impact if exploited:**  * Attacks that obtain access to a system and are allowed to look around in illegal locations unnoticed can do great harm to the organization. * When an organization experiences a data breach, it attracts criticism. Even after security breaches are fixed, users start to see them as unreliable or unsafe, which makes them less likely to accept personal information from them.  1. **How to fix the defect:**  * Organizations must have appropriate security controls in place to avoid the occurrence of sensitive data exposures as well as to limit their impacts on data subjects. * Organizations must have an effective breach response mechanism in place to immediately respond to sensitive data exposure.  1. **Authoritative reference links, and explanatory examples:**   Reference Link: <https://www.youtube.com/watch?v=2RKbacrkUBU>  Reference Link: <https://www.youtube.com/watch?v=l0YsEk_59fQ>  Reference Link: <https://portswigger.net/support/using-burp-to-test-for-sensitive-data-exposure-issue>  **Phase 2:**  **Owasp zap:**  Owasp zap is a penetration testing tool helps in finding vulnerabilities before  an attacker does.  **Step 1:**   * Open Owasp zap tool and click on “Automated Scan”.     **Step 2:**   * Paste the website link here that you want to automate scan.     **Step 3:**   * Select the preferences and click on “Attack”.     **We found these 11 vulnerabilities by doing an automated scan that will affect the web application:**     1. **Content Security Policy (CSP) Header Not Set:**   A Content Protection Policy (CSP) is a security standard that provides an additional layer of protection from cross-site scripting (XSS), and other code injection attacks. It is a defensive measure against any attacks that rely on executing malicious content in a trusted web context, or other attempts to circumvent the same-origin policy.     1. **Risk/ Undesirable impact if exploited:**   Hackers use XSS attacks to trick trusted websites into delivering malicious content. The browser executes all code from trusted origin and can’t differentiate between legitimate and malicious code, so any injected code is executed as well.   1. **How to fix the defect:**   To fix Content Security Policy (CSP) Header Not Set you need to configure your web server to return the Content-Security-Policy HTTP Header and giving it values to control what resources the browser is allowed to load for your page.   1. **Authoritative reference links, and explanatory examples:**   Reference Link: <https://www.youtube.com/watch?v=J90t0h0AP1U>   1. **Session ID in URL Rewrite:**   URL rewrite is used to track user session ID. The session ID may be disclosed via cross-site referer header. In addition, the session ID might be stored in browser history or server logs.     1. **Risk/ Undesirable impact if exploited:**   The Session Tokens (Cookie, SessionID, Hidden Field), if exposed, will usually enable an attacker to impersonate a victim and access the application.   1. **How to fix the defect:**  * Ensure using HTTPS on the website. * Store session ID in a cookie.  1. **Authoritative reference links, and explanatory examples:**   Reference Link: <https://www.youtube.com/watch?v=aDdQbi8dlTI>  Reference Link: <https://www.iothreat.com/blog/session-id-in-url-rewrite>   1. **Cross-Domain JavaScript Source File Inclusion:**   Cross-domain JavaScript source file inclusion is a security warning that can affect a web application that runs one or more Javascript files from a third-party domain. If the third-party intentionally or unintentionally holds a malicious content, it can be added and executed on the victim’s web application. This possibility occurs when the external Javascript is not validated. It can lead to the leakage of user data.     1. **Risk/ Undesirable impact if exploited:**  * Possible execution of malicious javascript. * Possible user data manipulation and leakage. * Malware infection.  1. **How to fix the defect:**  * Avoid placing sensitive information inside javascript files or JSONP. * Always try to sanitize user entries that are stored in JSON files.  1. **Authoritative reference links, and explanatory examples:**   Reference Link: <https://www.youtube.com/watch?v=X3YxJFiTa6c>  Reference Link: <https://www.iothreat.com/blog/cross-domain-javascript-source-file-inclusion>   1. **Private IP Disclosure:**   The IP address is a numerical label assigned to each device in a network. These numbers are used to uniquely identify devices in a network.  There are many servers that disclose the IP addresses of its users. This disclosure may leak information about the IP addressing scheme of a company’s/organisation’s internal network. An attacker can use the IP addresses to conduct further attacks on specific users.     1. **Risk/ Undesirable impact if exploited:**  * Network Layer attacks. * Possible loss of sensitive information.  1. **How to fix the defect:**  * Do not disclose the internal IP addresses. * Hide the private Ips in error messages. * Prevent the application from displaying the IP addresses of its users.  1. **Authoritative reference links, and explanatory examples:**   Reference Link: <https://whatismyipaddress.com/private-ip>   1. **Timestamp disclosure:**   Timestamp is a sequence of information that has been encoded to help in the identification of the time an event will occur. A timestamp disclosed by the application server or web server can be used to get other sensitive information.     1. **Risk/ Undesirable impact if exploited:**   If the server timestamp is used as a salt to hash specific sensitive information for authentication code, password, anti-CSRF token. But the attacker can retrieve it from the server. So the attacker can synchronize the local attacking code to minimize the number of brute force. So the attempts required to reproduce the result of the application hashing algorithm.   1. **How to fix the defect:**   Any Timestamp Disclosure alerts should be manually evaluated to verify that these are actual server timestamp leaks. But the disclosed timestamp data is not sensitive and it is not used in any form to generate any sensitive information on the server side.   1. **Authoritative reference links, and explanatory examples:**   Reference Link: <https://www.zaproxy.org/docs/alerts/10096/>   1. **Information Disclosure – Suspicious Comments:**   The information response appears to contain suspicious comments which may help an attacker.     1. **Risk/ Undesirable impact if exploited:**   The attacker may collect additional information about your application by learning source code fragments that were commented out and the actual source code comments.   1. **How to fix the defect:**  * All public-facing source code should be reviewed for any remaining comments. * Remove comments which have sensitive information about the design/implementation of the application.  1. **Authoritative reference links, and explanatory examples:**   Reference Link: <https://www.iothreat.com/blog/information-disclosure-suspicious-comments> | | | | |
| Assumptions:   1. Attacking a website with OWASP TOP 10 vulnerabilities will not affect the website. 2. Opening links on websites that take us to other website will not affect us. 3. We can secure our websites by not accepting hashes or numbers as input. So that hackers can't attack our website. 4. We can secure our websites by avoiding script tags in input. So that hackers can't attack our website. | | | | |
| Project Diagrams: | | | | |
| Algorithms:  eBay Releases Dynamic Application Security Testing Proxy as Open Source | | | | |
| Outcome:   * **Testing an insecure web application by using the DAST tool and exposing vulnerabilities.** | | | | |
| Exceptions considered:   1. While submitting the script in the message box, it is showing blank. 2. The website can't store the messages or comments. | | | | |
| Enhancement Scope:  My internship has taught me a lot about my skill set and given me confidence in my own abilities. It has helped guide my career aspirations and will definitely help me in my future career choices. I also gained knowledge about different attacks and vulnerabilities which affects a web application. The knowledge I gained will help me to face the real world post-graduation. My experience has brought me closer to my goals and I am excited for what the future has to bring! | | | | |
| Link to Code and executable file:  <https://github.com/vijayvj00/DAST_Testing> | | | | |