# Module:- SECURITY CONCEPT
# (ZAP_Report)
# Name:-Prithviraj Nikam

**Assignment:**

## Scan any Vulnerable website: https://www.hackthissite.org/. You can find yourself vulnerable website and try to scan and generate the report your's only

**Site: http://hackthissite.org**

**Alert Details:**

### 1.Absence of Anti-CSRF Tokens
### No Anti-CSRF tokens were found in a HTML submission form

**Description**- A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL /form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a website has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.
CSRF attacks are effective in a number of situations, including:
* The victim has an active session on the target site.
* The victim is authenticated via HTTP auth on the target site.
* The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

**Method**- GET
**Phase**: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard.
Phase: Implementation
**Solution-** Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

**Phase:** Architecture and Design
Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330)
Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

- Note that this can be bypassed using XSS.
- Use the ESAPI Session Management control.
- This control includes a component for CSRF.
- Do not use the GET method for any request that triggers a state change.

**2.CSP: Wildcard Directive**
**Description**- Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

**Method-** GET
**Solution-** Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

**3.CSP: script-src unsafe-inline**
**Description**- Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for

everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

**Method-** GET
**URL-** https://hackthissite.org/hp.php
**Solution-** Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.

**4.Cross-Domain Misconfiguration**
**Description-** Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web serve

**Method-** GET
**URL-** https://hackthissite.org/hp.php
**Solution-** Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).

Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

**1.Hidden File Found**
**Description-** A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.

**Method-** GET
**URL-** https://hackthissite.org/._darcs
**Solution-** Consider whether or not the component is actually required in production, if it isn't then disable it. If it is then ensure access to it requires appropriate authentication and authorization, or limit exposure to internal systems or specific source IPs, etc.