nemasis

Nemasis DA Report (OWASP 2013)

http://php.testsparker.com/process.php

Result Overview:

A5 Security Misconfiguration

X-Frame-Options Header Not Set

Description:
X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.

Solution:
Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).

References:
http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx

Url	Parameter	Evidence
/?file=Generics/index.nsp	X-Frame-Options	
/?file=Generics/contact.nsp	X-Frame-Options	
/?file=Generics/about.nsp	X-Frame-Options	
/	X-Frame-Options	

X-Content-Type-Options Header Missing

Description:
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type.

Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Solution:
Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

References:
http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx
https://www.owasp.org/index.php/List_of_useful_HTTP_headers

Url	Parameter	Evidence
/?file=Generics/index.nsp	X-Content-Type-Options	
/?file=Generics/contact.nsp	X-Content-Type-Options	
/	X-Content-Type-Options	
/?file=Generics/about.nsp	X-Content-Type-Options	

Content Security Policy (CSP) Header Not Set

Description:
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including 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.

Solution:
Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header, to achieve optimal browser support: "Content-Security-Policy" for Chrome 25+, Firefox 23+ and Safari 7+, "X-Content-Security-Policy" for Firefox 4.0+ and Internet Explorer 10+, and "X-WebKit-CSP" for Chrome 14+ and Safari 6+.

References:
https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
https://www.owasp.org/index.php/Content_Security_Policy
http://www.w3.org/TR/CSP/
http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html
http://www.html5rocks.com/en/tutorials/security/content-security-policy/
http://caniuse.com/#feat=contentsecuritypolicy
http://content-security-policy.com/

Url	Parameter	Evidence
/?file=Generics/index.nsp		
/		
/?file=Generics/contact.nsp		

/?file=Generics/about.nsp

A8 Cross-Site Request Forgery (CSRF)

Absence of Anti-CSRF Tokens

Description:
No Anti-CSRF tokens were found in a HTML submission form. 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 web site 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.

Solution:
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 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). Note that this can be bypassed using XSS. 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. Phase: Implementation Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

References:
http://projects.webappsec.org/Cross-Site-Request-Forgery
http://cwe.mitre.org/data/definitions/352.html

Url	Parameter	Evidence
/?file=Generics/about.nsp		
/?file=Generics/contact.nsp		
/?file=Generics/index.nsp		