Vulnerability Scan Report

Scanned URL: http://testphp.vulnweb.com/

1. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/robots.txt

2. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/

3. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/sitemap.xml

4. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/

5. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/high

6. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/privacy.php

7. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/index.php

8. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/categories.php

9. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/AJAX/index.php

10. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/cart.php

11. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/artists.php

12. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/index.php

13. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/categories.php

14. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/AJAX/index.php

15. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/disclaimer.php

16. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/guestbook.php

17. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/cart.php

18. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/

19. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/artists.php

20. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/

21. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/guestbook.php

22. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/disclaimer.php

23. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/hpp/

24. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/login.php

25. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/hpp/

26. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/listproducts.php?cat=2

27. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/listproducts.php?cat=4

28. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/search.php?test=query

29. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1

30. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/login.php

31. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/hpp/?pp=12

32. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=2

33. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/hpp/?pp=12

34. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/

35. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=4

36. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/search.php?test=query

37. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1

38. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/listproducts.php?cat=3

39. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/

40. Absence of Anti-CSRF Tokens

Severity: Medium

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.

URL: http://testphp.vulnweb.com/search.php?test=query

41. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=3

42. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/

43. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/

44. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/artists.php?artist=1

45. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/

46. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/signup.php

47. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/

48. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/artists.php?artist=3

49. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/artists.php?artist=1

50. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/artists.php?artist=2

51. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/signup.php

52. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/artists.php?artist=3

53. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/artists.php?artist=2

54. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/guestbook.php

55. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/product.php?pic=1

56. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/guestbook.php

57. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/product.php?pic=6

58. Absence of Anti-CSRF Tokens

Severity: Medium

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.

URL: http://testphp.vulnweb.com/guestbook.php

59. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/product.php?pic=1

60. Absence of Anti-CSRF Tokens

Severity: Medium

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.

URL: http://testphp.vulnweb.com/guestbook.php

61. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/product.php?pic=6

62. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/product.php?pic=5

63. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/product.php?pic=2

64. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/product.php?pic=5

65. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/product.php?pic=2

66. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/product.php?pic=4

67. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/product.php?pic=3

68. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12

69. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/product.php?pic=4

70. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/product.php?pic=3

71. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/product.php?pic=7

72. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12

73. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/product.php?pic=7

74. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/

75. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html

76. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/

77. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html

78. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/

79. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html

80. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/

81. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/

82. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html

83. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html

84. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/

85. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/listproducts.php?artist=1

86. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html

87. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/listproducts.php?artist=3

88. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/listproducts.php?artist=1

89. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/listproducts.php?artist=2

90. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/cart.php

91. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/listproducts.php?artist=3

92. Missing Anti-clickjacking Header

Severity: Medium

Description: The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. 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. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

URL: http://testphp.vulnweb.com/secured/newuser.php

93. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/listproducts.php?artist=2

94. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/cart.php

95. Content Security Policy (CSP) Header Not Set

Severity: Medium

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.

URL: http://testphp.vulnweb.com/secured/newuser.php

96. Absence of Anti-CSRF Tokens

Severity: Medium

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.

URL: http://testphp.vulnweb.com/cart.php

97. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/sitemap.xml

98. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/robots.txt

99. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your

web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/

100. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/

101. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/

102. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/high

103. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/style.css

104. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/privacy.php

105. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/userinfo.php

106. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/privacy.php

107. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/style.css

108. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/userinfo.php

109. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/index.php

110. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/categories.php

111. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/index.php

112. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/cart.php

113. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/categories.php

114. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/index.php

115. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/

116. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/cart.php

117. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/AJAX/index.php

118. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/artists.php

119. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other

frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/categories.php

120. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/guestbook.php

121. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/disclaimer.php

122. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/cart.php

123. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/

124. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/AJAX/index.php

125. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/artists.php

126. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/guestbook.php

127. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/disclaimer.php

128. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/

129. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/images/logo.gif

130. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components

may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/AJAX/index.php

131. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/artists.php

132. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/guestbook.php

133. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/disclaimer.php

134. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/hpp/

135. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/images/logo.gif

136. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Flash/add.swf

137. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/hpp/

138. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Flash/add.swf

139. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/hpp/

140. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/login.php

141. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/login.php

142. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/hpp/?pp=12

143. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/listproducts.php?cat=4

144. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/login.php

145. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/hpp/?pp=12

146. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/listproducts.php?cat=2

147. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/

148. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=4

149. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/images/remark.gif

150. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your

web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/search.php?test=query

151. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1

152. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/hpp/?pp=12

153. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/listproducts.php?cat=4

154. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/

155. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=2

156. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/images/remark.gif

157. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1

158. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/listproducts.php?cat=3

159. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/

160. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/listproducts.php?cat=2

161. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1

162. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/listproducts.php?cat=3

163. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/AJAX/styles.css

164. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/listproducts.php?cat=3

165. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/search.php?test=query

166. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/

167. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/AJAX/styles.css

168. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/

169. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/search.php?test=query

170. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/

171. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL:
http://testphp.vulnweb.com/showimage.php?file='%20+%20pict.item(0).firstChild.nodeValue%20+%20'

172. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/

173. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/1.jpg

174. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/

175. X-Content-Type-Options Header Missing

Severity: Low

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.

URL:
http://testphp.vulnweb.com/showimage.php?file='%20+%20pict.item(0).firstChild.nodeValue%20+%20'

176. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/

177. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/artists.php?artist=1

178. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/signup.php

179. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/1.jpg

180. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL:
http://testphp.vulnweb.com/showimage.php?file='%20+%20pict.item(0).firstChild.nodeValue%20+%20'

181. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/2.jpg

182. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/signup.php

183. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/artists.php?artist=1

184. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/3.jpg

185. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/artists.php?artist=3

186. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/2.jpg

187. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/signup.php

188. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/artists.php?artist=1

189. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/3.jpg

190. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/artists.php?artist=2

191. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/artists.php?artist=3

192. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg

193. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg

194. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/userinfo.php

195. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/artists.php?artist=2

196. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/artists.php?artist=3

197. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg

198. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg

199. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/userinfo.php

200. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/artists.php?artist=2

201. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg

202. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg

203. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/product.php?pic=1

204. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg

205. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg

206. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/guestbook.php

207. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/product.php?pic=1

208. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg

209. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/product.php?pic=6

210. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg&size;=160

211. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg

212. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/product.php?pic=1

213. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg

214. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/product.php?pic=6

215. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg&size;=160

216. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/guestbook.php

217. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg

218. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/product.php?pic=6

219. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg&size;=160

220. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your

web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/product.php?pic=5

221. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/guestbook.php

222. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/product.php?pic=2

223. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg

224. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg

225. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/product.php?pic=5

226. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/product.php?pic=2

227. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg

228. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg

229. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/product.php?pic=5

230. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/product.php?pic=2

231. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/product.php?pic=3

232. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg

233. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/product.php?pic=4

234. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12

235. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg

236. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/product.php?pic=3

237. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg&size;=160

238. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg&size;=160

239. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg&size;=160

240. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/product.php?pic=4

241. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12

242. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg&size;=160

243. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/product.php?pic=3

244. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg&size;=160

245. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg&size;=160

246. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/product.php?pic=4

247. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg&size;=160

248. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your

web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/product.php?pic=7

249. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12

250. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg&size;=160

251. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/2.jpg&size;=160

252. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg

253. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg&size;=160

254. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg&size;=160

255. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/product.php?pic=7

256. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg&size;=160

257. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg

258. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg&size;=160

259. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/product.php?pic=7

260. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg

261. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg&size;=160

262. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/

263. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html

264. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/

265. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html

266. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg&size;=160

267. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html

268. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/

269. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/

270. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/

271. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html

272. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html

273. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/

274. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html

275. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/

276. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/

277. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html

278. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html

279. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/

280. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg&size;=160

281. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html

282. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg&size;=160

283. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/listproducts.php?artist=1

284. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/listproducts.php?artist=3

285. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg&size;=160

286. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/listproducts.php?artist=1

287. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/listproducts.php?artist=3

288. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/listproducts.php?artist=1

289. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/listproducts.php?artist=3

290. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/listproducts.php?artist=2

291. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/secured/newuser.php

292. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/cart.php

293. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/listproducts.php?artist=2

294. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/secured/newuser.php

295. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/cart.php

296. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/listproducts.php?artist=2

297. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/secured/newuser.php

298. Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)

Severity: Low

Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

URL: http://testphp.vulnweb.com/cart.php

299. Server Leaks Version Information via "Server" HTTP Response Header Field

Severity: Low

Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.

Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

URL: http://testphp.vulnweb.com/secured/style.css

300. X-Content-Type-Options Header Missing

Severity: Low

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.

URL: http://testphp.vulnweb.com/secured/style.css

301. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/

302. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/index.php

303. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/AJAX/index.php

304. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/categories.php

305. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/cart.php

306. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/artists.php

307. Information Disclosure - Suspicious Comments

Severity: Informational

Description: The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.

Solution: Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

URL: http://testphp.vulnweb.com/AJAX/index.php

308. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/guestbook.php

309. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/disclaimer.php

310. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/AJAX/index.php

311. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/artists.php

312. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/login.php

313. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable

content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/listproducts.php?cat=2

314. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/listproducts.php?cat=4

315. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/search.php?test=query

316. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1

317. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/listproducts.php?cat=2

318. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1

319. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/listproducts.php?cat=3

320. User Controllable HTML Element Attribute (Potential XSS)

Severity: Informational

Description: This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.

Solution: Validate all input and sanitize output it before writing to any HTML attributes.

URL: http://testphp.vulnweb.com/search.php?test=query

321. User Controllable HTML Element Attribute (Potential XSS)

Severity: Informational

Description: This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.

Solution: Validate all input and sanitize output it before writing to any HTML attributes.

URL: http://testphp.vulnweb.com/search.php?test=query

322. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable

content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/artists.php?artist=1

323. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/signup.php

324. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/artists.php?artist=3

325. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/artists.php?artist=1

326. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/artists.php?artist=2

327. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/artists.php?artist=3

328. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/artists.php?artist=2

329. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/guestbook.php

330. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/product.php?pic=1

331. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/product.php?pic=6

332. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/product.php?pic=5

333. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/product.php?pic=2

334. User Controllable HTML Element Attribute (Potential XSS)

Severity: Informational

Description: This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.

Solution: Validate all input and sanitize output it before writing to any HTML attributes.

URL: http://testphp.vulnweb.com/guestbook.php

335. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/product.php?pic=4

336. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/product.php?pic=3

337. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/product.php?pic=7

338. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/listproducts.php?artist=1

339. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable

content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/listproducts.php?artist=3

340. Authentication Request Identified

Severity: Informational

Description: The given request has been identified as an authentication request. The 'Other Info' field contains a set of key=value lines which identify any relevant fields. If the request is in a context which has an Authentication Method set to "Auto-Detect" then this rule will change the authentication to match the request identified.

Solution: This is an informational alert rather than a vulnerability and so there is nothing to fix.

URL: http://testphp.vulnweb.com/secured/newuser.php

341. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/listproducts.php?artist=1

342. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and

manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/listproducts.php?artist=2

343. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/cart.php

344. Charset Mismatch (Header Versus Meta Content-Type Charset)

Severity: Informational

Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.

Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

URL: http://testphp.vulnweb.com/secured/newuser.php

345. Modern Web Application

Severity: Informational

Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.

Solution: This is an informational alert and so no changes are required.

URL: http://testphp.vulnweb.com/listproducts.php?artist=2