Valnara Security Scan Report

Scan Details:

Target URL: http://testphp.vulnweb.com/
Scan Type: Passive Scan
Scan Depth: 5
Start Time: 2025-04-09 18:04:17
End Time: 2025-04-09 18:05:30

Risk Summary:

High Risk Vulnerabilities: 12
Medium Risk Vulnerabilities: 76
Low Risk Vulnerabilities: 106
Informational Risk Vulnerabilities: 208

Detailed Vulnerabilities:

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/robots.txt
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/AJAX/index.php
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/hpp/
Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/high
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/images/logo.gif
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/userinfo.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/style.css
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/high
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/categories.php
Remediation: 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.

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

Risk Level: Low
URL: http://testphp.vulnweb.com/robots.txt
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Content Security Policy (CSP) Header Not Set

Risk Level: Medium
URL: http://testphp.vulnweb.com/hpp/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/disclaimer.php
Remediation: 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.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: 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.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/index.php
Remediation: 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.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/login.php
Remediation: 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.

X-Content-Type-Options Header Missing

Risk Level: Low
URL: http://testphp.vulnweb.com/images/logo.gif
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/cart.php
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/userinfo.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX/index.php
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/privacy.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/style.css
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/artists.php
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/AJAX/index.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/categories.php
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/disclaimer.php
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Flash/add.swf
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/login.php

Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/cart.php
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/index.php
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/privacy.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=4
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/
Remediation: 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.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/artists.php
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/categories.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Modern Web Application
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX/index.php
Remediation: This is an informational alert and so no changes are required.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/disclaimer.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/login.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/cart.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/index.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

X-Content-Type-Options Header Missing

Risk Level: Low
URL: http://testphp.vulnweb.com/Flash/add.swf
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/privacy.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Remediation: 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.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/listproducts.php?cat=4
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/artists.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low

URL: http://testphp.vulnweb.com/AJAX/index.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Absence of Anti-CSRF Tokens

Risk Level: Medium
URL: http://testphp.vulnweb.com/categories.php
Remediation: 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.

Absence of Anti-CSRF Tokens

Risk Level: Medium
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: 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.

Absence of Anti-CSRF Tokens

Risk Level: Medium
URL: http://testphp.vulnweb.com/disclaimer.php
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/login.php
Remediation: 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.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/cart.php
Remediation: 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.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/index.php
Remediation: 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.

Content Security Policy (CSP) Header Not Set

Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: 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.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=3
Remediation: 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.

Content Security Policy (CSP) Header Not Set

Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=4
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Absence of Anti-CSRF Tokens

Risk Level: Medium
URL: http://testphp.vulnweb.com/artists.php
Remediation: 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.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: 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.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/AJAX/index.php
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/categories.php

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

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/disclaimer.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/login.php
Remediation: 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.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/index.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/cart.php

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

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/listproducts.php?cat=3
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=4
Remediation: 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.

Modern Web Application
Risk Level: Informational
URL: http://testphp.vulnweb.com/artists.php
Remediation: This is an informational alert and so no changes are required.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/AJAX/index.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/disclaimer.php

Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/categories.php
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/index.php
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/cart.php
Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=3
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/artists.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/login.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Modern Web Application
Risk Level: Informational
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: This is an informational alert and so no changes are required.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=4
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

User Controllable HTML Element Attribute (Potential XSS)
Risk Level: Informational
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Validate all input and sanitize output it before writing to any HTML attributes.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/disclaimer.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/categories.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/index.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/cart.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/artists.php
Remediation: 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.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=3
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/login.php
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: 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.

User Controllable HTML Element Attribute (Potential XSS)
Risk Level: Informational
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Validate all input and sanitize output it before writing to any HTML attributes.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=4
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/guestbook.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/artists.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/login.php
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=4
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: 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.

Modern Web Application
Risk Level: Informational
URL: http://testphp.vulnweb.com/listproducts.php?cat=2

Remediation: This is an informational alert and so no changes are required.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=3
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=1
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=3
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=3
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/2.jpg
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/hpp/?pp=12
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/3.jpg
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/1.jpg
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/AJAX/styles.css
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Missing Anti-clickjacking Header
Risk Level: Medium

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

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Remediation: 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.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/2.jpg
Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/1.jpg
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/AJAX/styles.css
Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images/3.jpg
Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/hpp/?pp=12

Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set

Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set

Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set

Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set

Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Remediation: 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.

Missing Anti-clickjacking Header

Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Remediation: 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.

Missing Anti-clickjacking Header

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

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/?pp=12
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/?pp=12
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html

Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
Remediation: 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.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/

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

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/?pp=12
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Remediation: 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.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

X-Content-Type-Options Header Missing

Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Remediation: 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.

X-Content-Type-Options Header Missing
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12
Remediation: 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.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)
Risk Level: Low
URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=12
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

Cross Site Scripting (Reflected)
Risk Level: High
URL: http://testphp.vulnweb.com/search.php?test=%27%22%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E
Remediation: 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. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When

an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Cross Site Scripting (Reflected)
Risk Level: High
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: 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. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the

session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Cross Site Scripting (Reflected)

Risk Level: High
URL: http://testphp.vulnweb.com/hpp/params.php?p=%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E&pp;=12
Remediation: 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. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete

solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Cross Site Scripting (Reflected)
Risk Level: High
URL: http://testphp.vulnweb.com/listproducts.php?cat=%3CscrIpt%3Ealert%281%29%3B%3C%2FscrIpt%3E
Remediation: 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. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good"

input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Cross Site Scripting (Reflected)
Risk Level: High
URL: http://testphp.vulnweb.com/hpp/params.php?p=valid&pp;=%3CscrIpt%3Ealert%281%29%3B%3C%2FscRipt%3E
Remediation: 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. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list).

However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Cross Site Scripting (Reflected)

Risk Level: High
URL: http://testphp.vulnweb.com/hpp/?pp=%22+onMouseOver%3D%22alert%281%29%3B
Remediation: 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. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use an allow list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a deny list). However, deny lists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules.

As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

SQL Injection - MySQL
Risk Level: High
URL: http://testphp.vulnweb.com/search.php?test=%27
Remediation: Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.

SQL Injection - MySQL
Risk Level: High
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.

SQL Injection - MySQL
Risk Level: High
URL: http://testphp.vulnweb.com/listproducts.php?cat=%27
Remediation: Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.

SQL Injection - MySQL

Risk Level: High
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.

GET for POST
Risk Level: Informational
URL: http://testphp.vulnweb.com/search.php?test=query
Remediation: Ensure that only POST is accepted where POST is expected.

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/high
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/high
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/high
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/high
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Flash
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/AJAX
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/high
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/high
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/high
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/robots.txt
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/robots.txt
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/robots.txt
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/robots.txt
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/robots.txt
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/robots.txt
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/sitemap.xml
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/robots.txt
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/guestbook.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/hpp
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/images
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

User Agent Fuzzer
Risk Level: Informational
URL: http://testphp.vulnweb.com/userinfo.php
Remediation:

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/color-printer/3
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/web-camera-a4tech/2
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

Server Leaks Version Information via "Server" HTTP Response Header Field
Risk Level: Low
URL: http://testphp.vulnweb.com/Mod_Rewrite_Shop/Details/network-attached-storage-dlink/1

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

SQL Injection - MySQL
Risk Level: High
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.

Missing Anti-clickjacking Header
Risk Level: Medium
URL: http://testphp.vulnweb.com/
Remediation: 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.

Charset Mismatch (Header Versus Meta Content-Type Charset)
Risk Level: Informational
URL: http://testphp.vulnweb.com/
Remediation: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.

Content Security Policy (CSP) Header Not Set
Risk Level: Medium
URL: http://testphp.vulnweb.com/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Absence of Anti-CSRF Tokens
Risk Level: Medium
URL: http://testphp.vulnweb.com/
Remediation: 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.

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

Risk Level: Low
URL: http://testphp.vulnweb.com/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.

X-Content-Type-Options Header Missing

Risk Level: Low
URL: http://testphp.vulnweb.com/
Remediation: 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.

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

Risk Level: Low
URL: http://testphp.vulnweb.com/
Remediation: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.

SQL Injection - MySQL

Risk Level: High
URL: http://testphp.vulnweb.com/listproducts.php?cat=2
Remediation: Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.

Test Vulnerability

Risk Level: Low
URL: http://testphp.vulnweb.com/
Remediation: This is a test vulnerability added to verify the results display is working.