Security Scan Report

Injection Vulnerability: http://testphp.vulnweb.com/artists.php?artist=1"
Solution: Implement input validation and use parameterized queries to prevent SQL injection.

Insecure Design Vulnerability: No Insecure Design vulnerability detected.
Solution: Review and redesign the application architecture to address insecure design patterns.

Broken Access Control: Broken Access Control vulnerability detected at link: http://www.eclectasy.com/Fractal-Explorer/index.html
Solution: Implement proper access controls and enforce authorization mechanisms.

Security Misconfiguration: Security Misconfiguration vulnerability detected: Directory listing may be enabled for .git.
Solution: Review and configure server and application settings to eliminate security misconfigurations.

Sensitive Data Exposure: No Sensitive Data Exposure vulnerability detected.
Solution: Encrypt sensitive data in transit and at rest, and implement access controls to limit exposure.

Integrity Failure: No Integrity failure vulnerability detected.
Solution: Implement data integrity checks and validation mechanisms to ensure data consistency and prevent tampering.

Logging Monitoring Failure: TEST and Demonstration site for Acunetix Web Vulnerability Scanner home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo artist: r4w8173 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenatis. Aliquam posuere lobortis pede. Nullam fringilla urna id leo. Praesent aliquet pretium erat. Praesent non odio. Pellentesque a magna a mauris vulputate lacinia. Aenean viverra. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aliquam lacus. Mauris magna eros, semper a, tempor et, rutrum et, tortor. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin. Vestibulum condimentum facilisis nulla. In hac habitasse platea dictumst. Nulla nonummy. Cras quis libero. Cras venenatis. Aliquam posuere lobortis pede. Nullam fringilla urna id leo. Praesent aliquet pretium erat. Praesent non odio. Pellentesque a magna a mauris vulputate lacinia. Aenean viverra. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aliquam lacus. Mauris magna eros, semper a, tempor et, rutrum et, tortor. view pictures of the artistcomment on this artist search art Browse categories Browse artists Your cart Signup Your profile Our guestbook AJAX Demo Links Security art PHP scanner PHP vuln help Fractal Explorer About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more. Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking

skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.
Solution: Implement comprehensive logging and monitoring solutions to detect and respond to security incidents.

Server-Side Request Forgery (SSRF): SSRF vulnerability detected: http://testphp.vulnweb.com/artists.php?artist=1 contains sensitive data (Bank account number)
Solution: Validate and sanitize user-supplied input, and restrict access to sensitive resources.

Vulnerable Components: No Vulnerable components vulnerability detected.
Solution: Regularly update and patch software components to address known vulnerabilities.

Authentication Failure: Potential OWASP A07 vulnerability found (Login Form)
Solution: Implement secure authentication mechanisms such as multi-factor authentication and strong password policies.

Brute Force Vulnerability: No Brute force vulnerability detected.

Session Management Vulnerability: No Session management vulnerability detected.

API security check: The API endpoint http://testphp.vulnweb.com/artists.php?artist=1 is missing security headers: ['X-Content-Type-Options', 'X-Frame-Options', 'Content-Security-Policy']

OWASP API Security Check: The API endpoint http://testphp.vulnweb.com/artists.php?artist=1 is missing security headers: ['X-Content-Type-Options', 'X-Frame-Options', '
Content-Security-Policy']

ZAP Scan Results:

Issue 1:
Description: The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.
URL: http://testphp.vulnweb.com/artists.php?artist=1
Tags: OWASP_2021_A05, WSTG-v42-CLNT-09, OWASP_2017_A06
Risk: Medium
Solution: Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.
Reference: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

Issue 2:
Description: This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set. An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text.
URL: http://testphp.vulnweb.com/artists.php?artist=1
Tags:

Risk: Informational
Solution: Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.
Reference: https://code.google.com/p/browsersec/wiki/Part2#Character_set_handling_and_detection

Issue 3:
Description: Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL: http://testphp.vulnweb.com/artists.php?artist=1
Tags: OWASP_2021_A05, OWASP_2017_A06
Risk: Medium
Solution: Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.
Reference: https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html https://www.w3.org/TR/CSP/ https://w3c.github.io/webappsec-csp/ https://web.dev/articles/csp https://caniuse.com/#feat=contentsecuritypolicy https://content-security-policy.com/

Issue 4:
Description: No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.
URL: http://testphp.vulnweb.com/artists.php?artist=1
Tags: OWASP_2021_A01, WSTG-v42-SESS-05, OWASP_2017_A05
Risk: Medium
Solution: Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. Phase: Implementation Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. Phase: Architecture and Design Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). Note that this can be bypassed using XSS. Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that

operation. Note that this can be bypassed using XSS. Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change. Phase: Implementation Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.
Reference: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html https://cwe.mitre.org/data/definitions/352.html

Issue 5:
Description: The application appears to be a modern web application. If you need to explore it automatically then the Ajax Spider may well be more effective than the standard one.
URL: http://testphp.vulnweb.com/artists.php?artist=1
Tags:
Risk: Informational
Solution: This is an informational alert and so no changes are required.
Reference:

Issue 6:
Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.
URL: http://testphp.vulnweb.com/artists.php?artist=1
Tags: OWASP_2021_A05, OWASP_2017_A06, WSTG-v42-INFO-02
Risk: Low
Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.
Reference: https://httpd.apache.org/docs/current/mod/core.html#servertokens
https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff648552(v=pandp.10)
https://www.troyhunt.com/shhh-dont-let-your-response-headers/

Issue 7:
Description: The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL: http://testphp.vulnweb.com/artists.php?artist=1
Tags: OWASP_2021_A05, OWASP_2017_A06
Risk: Low
Solution: Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Reference: https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85) https://owasp.org/www-community/Security_Headers

Issue 8:
Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.
URL: http://testphp.vulnweb.com/artists.php?artist=1

Tags: OWASP_2021_A01, WSTG-v42-INFO-08, OWASP_2017_A03
Risk: Low
Solution: Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.
Reference: https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework
https://www.troyhunt.com/2012/02/shhh-dont-let-your-response-headers.html