report 3 brute_force attack

Sites: https://tiles-cdn.prod.ads.prod.webservices.mozgcp.net http://127.0.0.1

Generated on Thu, 22 Aug 2024 12:01:44

ZAP Version: 2.15.0

ZAP is supported by the Crash Override Open Source Fellowship

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	5
Low	6
Informational	5

Alerts

Name	Risk Level	Number of Instances
Absence of Anti-CSRF Tokens	Medium	3
Application Error Disclosure	Medium	2
Content Security Policy (CSP) Header Not Set	Medium	2
HTTP to HTTPS Insecure Transition in Form Post	Medium	2
Missing Anti-clickjacking Header	Medium	2
Cookie No HttpOnly Flag	Low	2
Cookie without SameSite Attribute	Low	2
Information Disclosure - Debug Error Messages	Low	1
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	2
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	2
X-Content-Type-Options Header Missing	Low	2
Authentication Request Identified	Informational	1
Cookie Poisoning	Informational	1
Information Disclosure - Suspicious Comments	Informational	6
Modern Web Application	Informational	2
User Controllable HTML Element Attribute (Potential XSS)	Informational	4

Alert Detail

Medium	Absence of Anti-CSRF Tokens

Description	No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL /form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf. CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via HTTP auth on the target site. * The victim is on the same local network as the target site. CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	<form action="https://www.paypal.com/cgi-bin/webscr" method="post" target="_blank">
Other Info	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "cmd" "hosted_button_id" "submit"].
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	<form action="https://www.paypal.com/cgi-bin/webscr" method="post" target="_blank">
Other Info	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "cmd" "hosted_button_id" "submit"].
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	<form action="index.php?page=login.php" method="post" enctype="application/x-www-form-urlencoded" onsubmit="return onSubmitOfLoginForm(this);" id="idLoginForm">
Other Info	No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 2: "login-php-submit-button" "password" "username"].
Instances	3
	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.

Solution	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
CWE Id	352
WASC Id	9
Plugin Id	10202

Medium	**Application Error Disclosure**
Description	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	<b>Warning</b>: "continue" targeting switch is equivalent to "break". Did you mean to use "continue 2"? in <b>C:\xampp\htdocs\mutillidae\owasp-esapi-php\lib\apache-log4php\trunk\src\main\php\helpers\LoggerPatternParser.php</b> on line <b>161</b><br />
Other Info	
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	<b>Warning</b>: "continue" targeting switch is equivalent to "break". Did you mean to use "continue 2"? in <b>C:\xampp\htdocs\mutillidae\owasp-esapi-php\lib\apache-log4php\trunk\src\main\php\helpers\LoggerPatternParser.php</b> on line <b>161</b><br />
Other Info	
Instances	2

Solution	Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.
Reference	
CWE Id	[200](200)
WASC Id	13
Plugin Id	[90022](90022)

Medium	**Content Security Policy (CSP) Header Not Set**
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	
Other Info	
Instances	2
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/
CWE Id	693
WASC Id	15
Plugin Id	10038

Medium	**HTTP to HTTPS Insecure Transition in Form Post**
Description	This check looks for insecure HTTP pages that host HTTPS forms. The issue is that an insecure HTTP page can easily be hijacked through MITM and the secure HTTPS form can be replaced or spoofed.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	

Evidence	https://www.paypal.com/cgi-bin/webscr
Other Info	The response to the following request over HTTP included an HTTPS form tag action attribute value: http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1 The context was: <form action="https://www.paypal.com/cgi-bin/webscr" method="post" target="_blank"> <input type="hidden" name="cmd" value="_s-xclick"> <input type="hidden" name="hosted_button_id" value="45R3YEXENU97S"> <input type="image" src="https://www.paypalobjects.com/en_US/i/btn/btn_donate_LG.gif" border="0" name="submit" alt="PayPal - The safer, easier way to pay online!"> <img alt="" border="0" src="https://www.paypalobjects.com/en_US/i/scr/pixel.gif" width="1" height="1"> </form>
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	https://www.paypal.com/cgi-bin/webscr
Other Info	The response to the following request over HTTP included an HTTPS form tag action attribute value: http://127.0.0.1/mutillidae/index.php?page=login.php The context was: <form action="https://www.paypal.com/cgi-bin/webscr" method="post" target="_blank"> <input type="hidden" name="cmd" value="_s-xclick"> <input type="hidden" name="hosted_button_id" value="45R3YEXENU97S"> <input type="image" src="https://www.paypalobjects.com/en_US/i/btn/btn_donate_LG.gif" border="0" name="submit" alt="PayPal - The safer, easier way to pay online!"> <img alt="" border="0" src="https://www.paypalobjects.com/en_US/i/scr/pixel.gif" width="1" height="1"> </form>
Instances	2
Solution	Use HTTPS for landing pages that host secure forms.
Reference	
CWE Id	[319](319)
WASC Id	15
Plugin Id	[10041](10041)

Medium	**Missing Anti-clickjacking Header**
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://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	
Other Info	
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	
Other Info	
Instances	2
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
CWE Id	1021
WASC Id	15
Plugin Id	10020

Low	**Cookie No HttpOnly Flag**
Description	A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	Set-Cookie: uid
Other Info	
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	Set-Cookie: username
Other Info	
Instances	2
Solution	Ensure that the HttpOnly flag is set for all cookies.
Reference	https://owasp.org/www-community/HttpOnly
CWE Id	1004
WASC Id	13
Plugin Id	10010

Low	**Cookie without SameSite Attribute**
Description	A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	Set-Cookie: uid
Other Info	
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	Set-Cookie: username
Other Info	
Instances	2
Solution	Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.

Reference	https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site
CWE Id	1275
WASC Id	13
Plugin Id	10054

Low	**Information Disclosure - Debug Error Messages**
Description	The response appeared to contain common error messages returned by platforms such as ASP.NET, and Web-servers such as IIS and Apache. You can configure the list of common debug messages.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	PHP error
Other Info	
Instances	1
Solution	Disable debugging messages before pushing to production.
Reference	
CWE Id	200
WASC Id	13
Plugin Id	10023

Low	**Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)**
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://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	X-Powered-By: PHP/8.2.12
Other Info	
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	X-Powered-By: PHP/8.2.12
Other Info	
Instances	2
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
CWE Id	200
WASC Id	13

Plugin Id	10037

Low	**Server Leaks Version Information via "Server" HTTP Response Header Field**
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://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Other Info	
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12
Other Info	
Instances	2
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/
CWE Id	200
WASC Id	13
Plugin Id	10036

Low	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	
Other Info	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	
Other	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages

Info	away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.
Instances	2
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
CWE Id	693
WASC Id	15
Plugin Id	10021

Informational	**Authentication Request Identified**
Description	The given request has been identified as an authentication request. The 'Other Info' field contains a set of key=value lines which identify any relevant fields. If the request is in a context which has an Authentication Method set to "Auto-Detect" then this rule will change the authentication to match the request identified.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	password
Other Info	userParam=login-php-submit-button userValue=Login passwordParam=password referer=https://127.0.0.1/mutillidae/index.php?page=login.php
Instances	1
Solution	This is an informational alert rather than a vulnerability and so there is nothing to fix.
Reference	https://www.zaproxy.org/docs/desktop/addons/authentication-helper/auth-req-id/
CWE Id	
WASC Id	
Plugin Id	10111

Informational	**Cookie Poisoning**
Description	This check looks at user-supplied input in query string parameters and POST data to identify where cookie parameters might be controlled. This is called a cookie poisoning attack, and becomes exploitable when an attacker can manipulate the cookie in various ways. In some cases this will not be exploitable, however, allowing URL parameters to set cookie values is generally considered a bug.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	
Other Info	An attacker may be able to poison cookie values through POST parameters. To test if this is a more serious issue, you should try resending that request as a GET, with the POST parameter included as a query string parameter. For example: https://nottrusted.com/page?value=maliciousInput. This was identified at: http://127.0.0.1/mutillidae/index.php?page=login.php User-input was found in the following cookie: username=admin The user input was: username=admin
Instances	1

Solution	Do not allow user input to control cookie names and values. If some query string parameters must be set in cookie values, be sure to filter out semicolon's that can serve as name/value pair delimiters.
Reference	https://en.wikipedia.org/wiki/HTTP_cookie https://cwe.mitre.org/data/definitions/565.html
CWE Id	565
WASC Id	20
Plugin Id	10029

Informational	**Information Disclosure - Suspicious Comments**
Description	The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	from
Other Info	The following pattern was used: \bFROM\b and was detected in the element starting with: "<script type="text/javascript"> try{ //if(!window.localStorage.length){ window.localStorage.setItem("SelfDestruct", see evidence field for the suspicious comment/snippet.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	from
Other Info	The following pattern was used: \bFROM\b and was detected in the element starting with: "<script type="text/javascript"> try{ //if(!window.localStorage.length){ window.localStorage.setItem("SelfDestruct", see evidence field for the suspicious comment/snippet.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	username
Other Info	The following pattern was used: \bUSERNAME\b and was detected 2 times, the first in the element starting with: "<script type="text/javascript"> <!-- var l_loggedIn = false; var lAuthenticationAttemptResultFlag = 0; var lValidateInput = '', see evidence field for the suspicious comment/snippet.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	username
Other Info	The following pattern was used: \bUSERNAME\b and was detected 2 times, the first in the element starting with: "<script type="text/javascript"> <!-- var l_loggedIn = false; var lAuthenticationAttemptResultFlag = 1; var lValidateInput = '', see evidence field for the suspicious comment/snippet.
URL	http://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	user
Other Info	The following pattern was used: \bUSER\b and was detected 2 times, the first in the element starting with: "<!-- I think the database password is set to blank or perhaps samurai. It depends on whether you installed this web app from ", see evidence field for the

	suspicious comment/snippet.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	user
Other Info	The following pattern was used: \bUSER\b and was detected in the element starting with: "<!-- I think the database password is set to blank or perhaps samurai. It depends on whether you installed this web app from ", see evidence field for the suspicious comment/snippet.
Instances	6
Solution	Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.
Reference	
CWE Id	200
WASC Id	13
Plugin Id	10027

Informational	**Modern Web Application**
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://127.0.0.1/mutillidae/index.php?popUpNotificationCode=AU1
Method	GET
Attack	
Evidence	<a href="">OWASP 2017</a>
Other Info	Links have been found that do not have traditional href attributes, which is an indication that this is a modern web application.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	<a href="">OWASP 2017</a>
Other Info	Links have been found that do not have traditional href attributes, which is an indication that this is a modern web application.
Instances	2
Solution	This is an informational alert and so no changes are required.
Reference	
CWE Id	
WASC Id	
Plugin Id	10109

Informational	**User Controllable HTML Element Attribute (Potential XSS)**
Description	This check looks at user-supplied input in query string parameters and POST data to identify where certain HTML attribute values might be controlled. This provides hot-spot detection for XSS (cross-site scripting) that will require further review by a security analyst to determine exploitability.
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	

Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1/mutillidae/index.php?page=login.php appears to include user input in: a(n) [input] tag [name] attribute The user input found was: login-php-submit-button=Login The user-controlled value was: login-php-submit-button
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1/mutillidae/index.php?page=login.php appears to include user input in: a(n) [input] tag [value] attribute The user input found was: login-php-submit-button=Login The user-controlled value was: login
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1/mutillidae/index.php?page=login.php appears to include user input in: a(n) [input] tag [name] attribute The user input found was: password=password The user-controlled value was: password
URL	http://127.0.0.1/mutillidae/index.php?page=login.php
Method	POST
Attack	
Evidence	
Other Info	User-controlled HTML attribute values were found. Try injecting special characters to see if XSS might be possible. The page at the following URL: http://127.0.0.1/mutillidae/index.php?page=login.php appears to include user input in: a(n) [input] tag [type] attribute The user input found was: password=password The user-controlled value was: password
Instances	4
Solution	Validate all input and sanitize output it before writing to any HTML attributes.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html
CWE Id	20
WASC Id	20
Plugin Id	10031