El Alto CLAMP Security/Vulnerability Report

This table represents the known exploitable and non-exploitable vulnerabilities in third party packages used in the project.

Repository	Group	Problem Code	Effective /Ineffective	Resolvable by Project	Impact Analysis	Action
clamp	com. fasterxml. jackson. core	N/A	Effective	No	`jackson-databind` is vulnerable to Remote Code Execution (RCE). The `createBeanDeserializer()` function in the `BeanDeserializerFactory` class allows untrusted Java objects to be deserialized. A remote attacker can exploit this by uploading a malicious serialized object that will result in RCE if the application attempts to deserialize it. Note: This vulnerability exists due to the incomplete fix for CVE-2017-7525, CVE-2017-15095, CVE-2017-17485, CVE-2018-5968, CVE-2018-7489, CVE-2018-11307, CVE-2018-12022, CVE-2018-12023, CVE-2018-14718, CVE-2018-14719, CVE-2018-14720, and CVE-2018-14721. Evidence of this can be found at https://pivotal.io/security/cve-2017-4995. The application is vulnerable by using this component, when default typing is enabled and passing in untrusted data to be deserialization.	the issue has been removed from the CLAMP core code. the remaining usage of "Jackson" is coming from sdc client library so we depend on SDC project to remove the final reference to "Jackson" library. ~~SDC-2216~~ - Security improvements CLOSED
clamp	com. fasterxml. jackson. core	CVE-2019-12086	Effective	No	A Polymorphic Typing issue was discovered in FasterXML jackson-databind 2.x before 2.9.9. When Default Typing is enabled (either globally or for a specific property) for an externally exposed JSON endpoint, the service has the mysql-connector-java jar (8.0.14 or earlier) in the classpath, and an attacker can host a crafted MySQL server reachable by the victim, an attacker can send a crafted JSON message that allows them to read arbitrary local files on the server. This occurs because of missing com. mysql.cj.jdbc.admin.MiniAdmin validation. The `jackson-databind` component contains an Insecure Deserialization Vulnerability. The `validateSubType()` function in the `SubTypeValidator` class allows untrusted input to be deserialized as a `com.mysql.cj.jdbc.admin.MiniAdmin` instance. A remote attacker can exploit this behavior by submitting a crafted JSON payload to a deserializing endpoint that uses `jackson-databind`. The attacker's deserialized Java object may then create a connection with an attacker controlled MySql server. When running a `mysql-connector-java` jar versioned 8.0.14 and earlier, the attacker's MySql server may read arbitrary files accessible by the vulnerable application when the connection is established. The application is vulnerable by using this component if a `mysql-connector-java` jar versioned 8.0.14 or lower is in the classpath while Default Typing is enabled.	the issue has been removed from the CLAMP core code. the remaining usage of "Jackson" is coming from sdc client library so we depend on SDC project to remove the final reference to "Jackson" library. ~~SDC-2216~~ - Security improvements CLOSED
clamp	jquery	N/A	Ineffective	Yes	The `jquery` package is vulnerable to Prototype Pollution. The `jQuery.extend` and `jQuery.fn.extend` functions defined in many files allow an untrusted object to extend `Object.prototype`. An attacker can modify and add prototype properties to JavaScript objects and can potentially leverage those changes to crash the application or execute remote code.	not effective in the new El Alto code
clamp	angular	N/A	Ineffective	Yes	The `angular` package in NPM, and the `AngularJS.Core` and `AngularJS` packages in NuGet are vulnerable to Cross Site Scripting (XSS). The `sendReq()` function in the `angular.js` file does not validate the `JSONP` endpoint URL against the predefined trusted resource URLs before processing the `JSONP` request. An attacker can exploit this vulnerability by enticing the victim to send a malicious `JSONP` request. Once the request is sent, malicious JavaScript is returned from the attacker's website, which is then executed in the victim's browser.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	angular	N/A	Ineffective	Yes	The `angular` package is vulnerable to Cross-Site Scripting (XSS). The `computedMember()`, `nonComputedMember()`, and `recurse()` functions in the `angular.js` file allows JavaScript to be injected in the constructor properties of an object in angular expressions. An attacker can exploit this vulnerability by injecting an expression crafted with malicious script assigned to the constructor property of an object which, when parsed, results in XSS.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	angular	N/A	Ineffective	Yes	AngularJS is vulnerable to Cross-Site Scripting (XSS). The `getTrustedContext()` function in `compile.js` allows malicious links through the `href` attribute of a `Link` element, as this element has no protection from the `$sce` module. An attacker can exploit this vulnerability by crafting input placed in the `href` attribute of a `Link` element to contain malicious script, which leads to Cross-Site Scripting.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED

clamp	angular	N/A	Ineffective	Yes	NO INFORMATION	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	angular	N/A	Ineffective	Yes	AngularJS is vulnerable to Cross-Site Scripting (XSS). The `$set` function in `compile.js` allows JavaScript in the `xlink:href` attribute of an anchor (a) element within the svg element without sanitizing it. An attacker can exploit this vulnerability by crafting the `xlink:href` attribute of an anchor (a) element with malicious script, that when rendered results in Cross-Site Scripting.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	angular	N/A	Ineffective	Yes	The `angular` package in NPM, and the `AngularJS.Core` and `AngularJS` packages in NuGet are vulnerable to Cross-Site Request Forgery (CSRF). The `sendReq()` function in the `angular.js` file doesn't properly validate the callback parameter. One likely exploit scenario involves the use of SWF files, deemed the Rosetta Flash vulnerability. This means that a remote attacker can host a page containing an `<object>` element with its `type` attribute set to “application/x-shockwave-flash” and its `data` attribute set to the vulnerable JSONP endpoint URL where the `callback` parameter of the URL is set to an alphanumeric encoding of a malicious SWF file. When rendered in the browser, the reflected SWF data will be recognized as a valid SWF file and will execute. Because the SWF file is reflected from the vulnerable site and may make cookie carrying requests, the browser believes the vulnerable site to be its origin, bypassing the Same Origin Policy. This allows making requests to the vulnerable site and possibly exfiltrating the vulnerable site's data to the attacker's domain.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	angular	N/A	Ineffective	Yes	AngularJS, when used in browser extensions, is vulnerable to a Content Security Policy (CSP) bypass vulnerability. The `angularInit()` function in the `angular.js` file does not properly prevent auto-bootstrapping when loaded from extensions. As a part of the AngularJS bootstrap process, portions of the HTML DOM are evaluated and potentially executed; by including AngularJS in a browser extension, it is possible to utilize this behavior to bypass the CSP of the target application. An attacker can leverage this vulnerability to execute XSS that would otherwise have been prevented by the CSP against victims that use a browser extension that includes AngularJS.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	angular	N/A	Ineffective	Yes	AngularJS, when used in browser extensions, is vulnerable to a Content Security Policy (CSP) bypass vulnerability. As a part of the AngularJS bootstrap process, portions of the HTML DOM are evaluated and potentially executed; by including AngularJS in a browser extension, it is possible to utilize this behavior to bypass the CSP of the target application. An attacker can leverage this vulnerability to execute XSS that would otherwise have been prevented by the CSP against victims that use a browser extension that includes AngularJS.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	angular	N/A	Ineffective	Yes	Angular is vulnerable to Cross-Site Scripting. The `sanitizeUri()` function in `angular.js` allows arbitrary JavaScript to be injected into certain HTML attributes in the form of `javascript:` protocol URIs. If user input is used to generate an HTML attribute containing a URI (such as an `hr ef` or `src` attribute), an attacker can exploit this vulnerability by injecting a URI containing malicious JavaScript.	not effective anymore in the new El Alto code. Old (angular) code will be removed from repository when Resources are available. ~~CLAMP-223~~ - replace "angular.js" and move to "React" for security issues CLOSED
clamp	commons-codec	N/A	Effective	No	The Apache `commons-codec` package contains an Improper Input Validation vulnerability. The `decode()` method in the `Base32`, `Base64`, and `BCodec` classes fails to reject malformed Base32 and Base64 encoded strings and consequently decodes them into arbitrary values. A remote attacker can leverage this vulnerability to potentially tunnel additional information via seemingly legitimate Base32 or Base64 encoded strings.	coming from the SDC client library so, it has to be solved by the SDC team. SDC doesn't have a Jira ticket yet for this issue. ~~SDC-2504~~ - Review commons-codec usage in SDC client library CLOSED

clamp	org.springframework.security	N/A	Ineffective	No	The `spring-security-web` package is vulnerable to Cross-Site Request Forgery (CSRF). The `doFilter()` method in the `SwitchUserFilter`, which is reachable via a GET request, does not require any form of confirmation that the user sending the request intended to do so. An attacker can exploit this vulnerability by crafting a malicious application containing links to the vulnerable endpoint, HTML tags that use the vulnerable endpoint in the `src` attribute, or malicious JavaScript designed to send the request to the vulnerable endpoint. When a victim visits the malicious page, their browser will be made to send requests to the vulnerable endpoint, taking action as the victim without the victim's knowledge or consent.	~~CLAMP-282~~ - spring-security-web vulnerability issue CLOSED