Fortify Security Report

Jan 22, 2020

pgupta25

Executive Summary

Issues Overview

On Jan 22, 2020, a source code review was performed over the owa code base. 100 files, 3,913 LOC (Executable) were scanned and reviewed for defects that could lead to potential security vulnerabilities. A total of 14 reviewed findings were uncovered during the analysis.

Issues by Fortify Priority Order	
Critical	8
High	6

Recommendations and Conclusions

The Issues Category section provides Fortify recommendations for addressing issues at a generic level. The recommendations for specific fixes can be extrapolated from those generic recommendations by the development group.

Project Summary

Code Base Summary

Code location: /srv/openmrs_code/org/openmrs/module/owa
Number of Files: 100
Lines of Code: 3913
Build Label: <No Build Label>

Scan Information

Scan time: 17:29
SCA Engine version: 19.1.0.2241
Machine Name: vclv99-89.hpc.ncsu.edu
Username running scan: pgupta25

Results Certification

Results Certification Valid

Details:

Results Signature:

SCA Analysis Results has Valid signature

Rules Signature:

There were no custom rules used in this scan

Attack Surface

Attack Surface:
File System:
java.io.FileInputStream.FileInputStream
java.io.FileInputStream.FileInputStream
java.util.zip.ZipFile.getEntry

Stream:
java.io.RandomAccessFile.read

System Information:
null.null.null
null.null.e
null.null.error
null.null.k
null.null.lambda
null.null.q
java.io.File.listFiles
java.lang.Throwable.getMessage
javax.servlet.ServletContext.getRealPath

Web:

javax.servlet.ServletRequest.getScheme

Filter Set Summary

Current Enabled Filter Set:
Quick View

Filter Set Details:

Folder Filters:
If [fortify priority order] contains critical Then set folder to Critical
If [fortify priority order] contains high Then set folder to High
If [fortify priority order] contains medium Then set folder to Medium
If [fortify priority order] contains low Then set folder to Low
Visibility Filters:
If impact is not in range [2.5, 5.0] Then hide issue
If likelihood is not in range (1.0, 5.0] Then hide issue

Audit Guide Summary

Audit guide not enabled

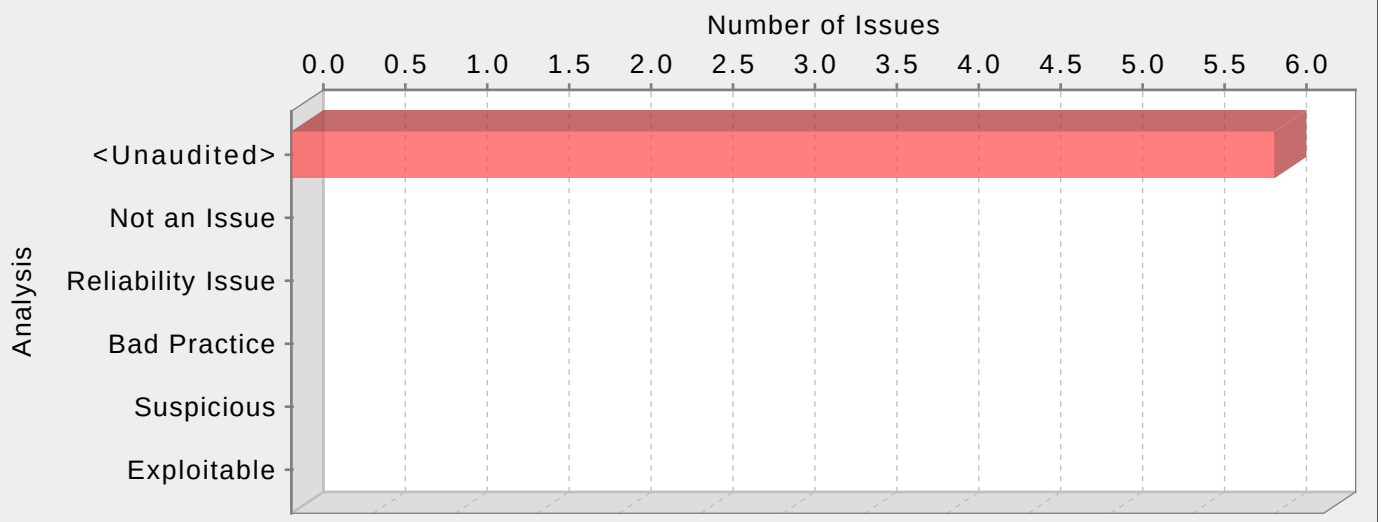
Results Outline

Overall number of results

The scan found 14 issues.

Vulnerability Examples by Category

Category: Cross-Site Scripting: DOM (6 Issues)



Abstract:

The method Aa() in angular-1.4.5.min.js sends unvalidated data to a web browser on line 146, which can result in the browser executing malicious code.

Explanation:

Cross-site scripting (XSS) vulnerabilities occur when:

1. Data enters a web application through an untrusted source. In the case of DOM-based XSS, data is read from a URL parameter or other value within the browser and written back into the page with client-side code. In the case of reflected XSS, the untrusted source is typically a web request, while in the case of persisted (also known as stored) XSS it is typically a database or other back-end data store.

2. The data is included in dynamic content that is sent to a web user without being validated. In the case of DOM Based XSS, malicious content gets executed as part of DOM (Document Object Model) creation, whenever the victim's browser parses the HTML page.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash or any other type of code that the browser executes. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Example 1: The following JavaScript code segment reads an employee ID, eid, from a URL and displays it to the user.

```
<SCRIPT>
var pos=document.URL.indexOf("eid=")+4;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
```

Example 2: Consider the HTML form:

```
<div id="myDiv">
Employee ID: <input type="text" id="eid"><br>
...
<button>Show results</button>
</div>
<div id="resultsDiv">
...
</div>
```

The following jQuery code segment reads an employee ID from the form, and displays it to the user.

```
$(document).ready(function(){
$("#myDiv").on("click", "button", function(){
var eid = $("#eid").val();
$("resultsDiv").append(eid);
...
});
});
```

These code examples operate correctly if the employee ID, from the text input with ID eid contains only standard alphanumeric text. If eid has a value that includes meta-characters or source code, then the code will be executed by the web browser as it displays the HTTP response.

Example 3: The following code shows an example of a DOM-based XSS within a React application:

```
let element = JSON.parse(getUntrustedInput());
ReactDOM.render(<App>
{element}
</App>);
```

In Example 3, if an attacker can control the entire JSON object retrieved from getUntrustedInput(), they may be able to make React render element as a component, and therefore can pass an object with dangerouslySetInnerHTML with their own controlled value, a typical cross-site scripting attack.

Initially these might not appear to be much of a vulnerability. After all, why would someone provide input containing malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use email or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web application back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

As the example demonstrates, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim:

- Data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or emailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that may include session information, from the user's machine to the attacker or perform other nefarious activities.

- The application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Persistent XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

- A source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

Recommendations:

The solution to XSS is to ensure that validation occurs in the correct places and checks are made for the correct properties.

Since XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS.

Web applications must validate their input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application may accept input through a shared data store or other trusted source, and that data store may accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means the best way to prevent XSS vulnerabilities is to validate everything that enters the application and leaves the application destined for the user.

The most secure approach to validation for XSS is to create a whitelist of safe characters that are allowed to appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alpha-numeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser should still be considered valid input once they are encoded, such as a web design bulletin board that must accept HTML fragments from its users.

A more flexible, but less secure approach is known as blacklisting, which selectively rejects or escapes potentially dangerous characters before using the input. In order to form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines what characters have special meaning, many web browsers try to correct common mistakes in HTML and may treat other characters as special in certain contexts, which is why we do not encourage the use of blacklists as a means to prevent XSS. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]:

In the content of a block-level element (in the middle of a paragraph of text):

- "<" is special because it introduces a tag.

- "&" is special because it introduces a character entity.

- ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error.

The following principles apply to attribute values:

- In attribute values enclosed with double quotes, the double quotes are special because they mark the end of the attribute value.

- In attribute values enclosed with single quote, the single quotes are special because they mark the end of the attribute value.

- In attribute values without any quotes, white-space characters, such as space and tab, are special.

- "&" is special when used with certain attributes, because it introduces a character entity.

In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters:

- Space, tab, and new line are special because they mark the end of the URL.

- "&" is special because it either introduces a character entity or separates CGI parameters.

- Non-ASCII characters (that is, everything greater than 127 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context.

- The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server -side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page in question.

Within the body of a <SCRIPT> </SCRIPT>:

- Semicolons, parentheses, curly braces, and new line characters should be filtered out in situations where text could be inserted directly into a pre-existing script tag.

Server-side scripts:

- Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering.

Other possibilities:

- If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and may bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7).

After you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option in this situation is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and may be unacceptable in circumstances where the integrity of the input must be preserved for display.

If input containing special characters must be accepted and displayed accurately, validation must encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2].

Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. When an application is developed there are no guarantees about what application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will also stay in sync.

Tips:

1. The Fortify Secure Coding Rulepacks warn about SQL Injection and Access Control: Database issues when untrusted data is written to a database and also treat the database as a source of untrusted data, which can lead to XSS vulnerabilities. If the database is a trusted resource in your environment, use custom filters to filter out dataflow issues that include the DATABASE taint flag or originate from database sources. Nonetheless, it is often still a good idea to validate everything read from the database.

2. Even though URL encoding untrusted data protects against many XSS attacks, some browsers (specifically, Internet Explorer 6 and 7 and possibly others) automatically decode content at certain locations within the Document Object Model (DOM) prior to passing it to the JavaScript interpreter. To reflect this danger, the rulepacks no longer treat URL encoding routines as sufficient to protect against cross-site scripting. Data values that are URL encoded and subsequently output will cause Fortify to report Cross-Site Scripting: Poor Validation vulnerabilities.

3. Older versions of React are more susceptible to cross-site scripting attacks by controlling an entire component. Newer versions use Symbols to identify a React component, which prevents the exploit, however older browsers that do not have Symbol support (natively, or through polyfills), such as all versions of Internet Explorer, are still vulnerable. Other types of cross-site scripting attacks are valid for all browsers and versions of React.

angular-1.4.5.min.js, line 146 (Cross-Site Scripting: DOM)

Fortify Priority:	Critical	**Folder**	Critical
Kingdom:	Input Validation and Representation		
Abstract:	The method Aa() in angular-1.4.5.min.js sends unvalidated data to a web browser on line 146, which can result in the browser executing malicious code.		

Source: angular-1.4.5.min.js:146 Read Z.href()

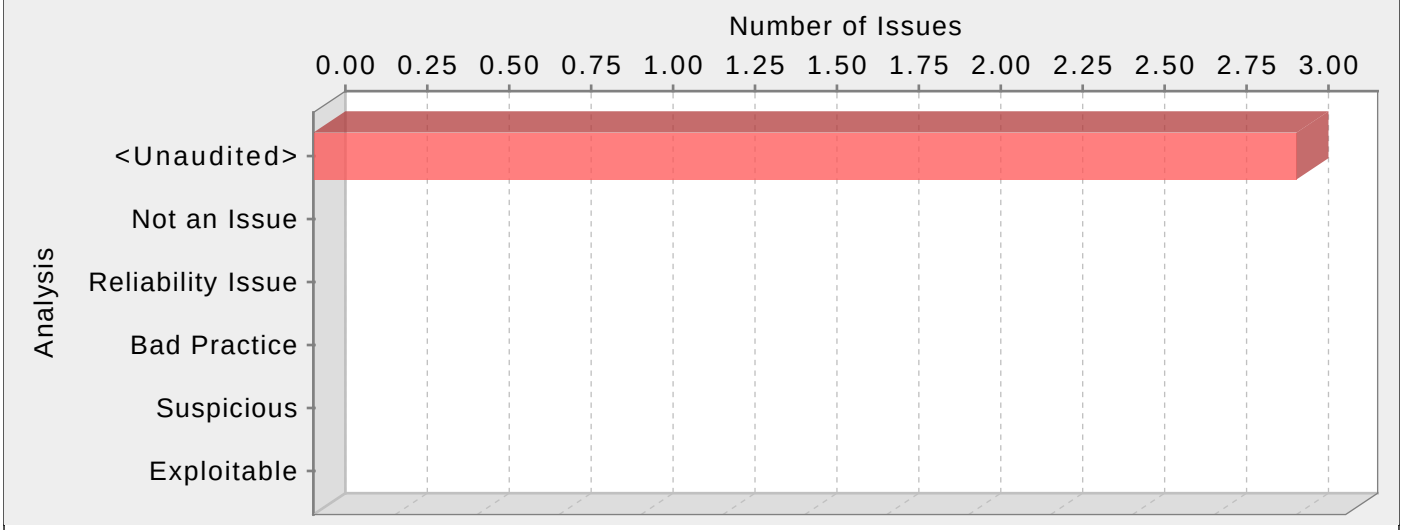
```
144  return e}]}function
     sf(){this.$get=["$rootScope","$browser","$location",function(b,a,c){return{findBinding
     s:function(a,b,c){a=a.getElementsByClassName("ng-binding");var
     g=[];n(a,function(a){var d=aa.element(a).data("$binding");d&&n(d,function(d){c?(new
     RegExp("(^|\\s)"+ud(b)+"(\\s|\\||$)")).test(d)&&g.push(a):-
     1!=d.indexOf(b)&&g.push(a)})});return g},findModels:function(a,b,c){for(var g=["ng-
     ","data-ng-","ng\\:"],h=0;h<g.length;++h){var
     l=a.querySelectorAll("["+g[h]+"model"+(c?"=":"*=")+'"'+b+'"...
145  if(l.length)return l}},getLocation:function(){return
     c.url()},setLocation:function(a){a!==c.url()&&(c.url(a),b.$digest())},whenStable:funct
     ion(b){a.notifyWhenNoOutstandingRequests(b)}}]}function
     tf(){this.$get=["$rootScope","$browser","$q","$$q","$exceptionHandler",function(b,a,c,
     d,e){function f(f,l,k){B(f)||(k=l,l=f,f=v);var
     m=xa.call(arguments,3),q=x(k)&&!k,s=(q?d:c).defer(),t=s.promise,n;n=a.defer(function()
     {try{s.resolve(f.apply(null,m))}catch(a){s.reject(a),e(a)}finally{delete
     g[t.$$timeou...
146  b.$apply()},l);t.$$timeoutId=n;g[n]=s;return t}var g={};f.cancel=function(b){return
     b&&b.$$timeoutId in g?(g[b.$$timeoutId].reject("canceled"),delete
     g[b.$$timeoutId],a.defer.cancel(b.$$timeoutId)):!1};return f}]}function
     Aa(b){Va&&(Z.setAttribute("href",b),b=Z.href);Z.setAttribute("href",b);return{href:Z.h
     ref,protocol:Z.protocol?Z.protocol.replace(/:$/,""):"",host:Z.host,search:Z.search?Z.s
     earch.replace(/^\?/,""):"",hash:Z.hash?Z.hash.replace(/^#/,""):"",hostname:Z.hostname,
     port:Z.port,pathname...
147  Z.pathname.charAt(0)?Z.pathname:"/"+Z.pathname}}function gd(b){b=H(b)?Aa(b):b;return
     b.protocol===wd.protocol&&b.host===wd.host}function uf(){this.$get=qa(N)}function
     xd(b){function a(a){try{return decodeURIComponent(a)}catch(b){return a}}var
     c=b[0]||{},d={},e="";return function(){var
     b,g,h,l,k;b=c.cookie||"";if(b!==e)for(e=b,b=e.split("
     "),d={},h=0;h<b.length;h++)g=b[h],l=g.indexOf("="),0<l&&(k=a(g.substring(0,l)),d[k]===
     u&&(d[k]=a(g.substring(l+1))));return d}}function yf(){this.$get=xd}funct...
148  d){if(D(c)){var e={};n(c,function(b,c){e[c]=a(c,b)});return e}return
     b.factory(c+"Filter",d)}this.register=a;this.$get=["$injector",function(a){return
     function(b){return
     a.get(b+"Filter")}}];a("currency",yd);a("date",zd);a("filter",$f);a("json",ag);a("limi
     tTo",bg);a("lowercase",cg);a("number",Ad);a("orderBy",Bd);a("uppercase",dg)}function
     $f(){return function(b,a,c){if(!Da(b)){if(null==b)return b;throw
     G("filter")("notarray",b);}var d;switch(gc(a)){case "function":break;case
     "boolean":case "null...
```

Sink: angular-1.4.5.min.js:146 ~JS_Generic.setAttribute()

```
144  return e}]}function
     sf(){this.$get=["$rootScope","$browser","$location",function(b,a,c){return{findBinding
     s:function(a,b,c){a=a.getElementsByClassName("ng-binding");var
     g=[];n(a,function(a){var d=aa.element(a).data("$binding");d&&n(d,function(d){c?(new
     RegExp("(^|\\s)"+ud(b)+"(\\s|\\||$)")).test(d)&&g.push(a):-
     1!=d.indexOf(b)&&g.push(a)})});return g},findModels:function(a,b,c){for(var g=["ng-
     ","data-ng-","ng\\:"],h=0;h<g.length;++h){var
     l=a.querySelectorAll("["+g[h]+"model"+(c?"=":"*=")+'"'+b+'"...
145  if(l.length)return l}},getLocation:function(){return
     c.url()},setLocation:function(a){a!==c.url()&&(c.url(a),b.$digest())},whenStable:funct
     ion(b){a.notifyWhenNoOutstandingRequests(b)}}]}function
     tf(){this.$get=["$rootScope","$browser","$q","$$q","$exceptionHandler",function(b,a,c,
     d,e){function f(f,l,k){B(f)||(k=l,l=f,f=v);var
     m=xa.call(arguments,3),q=x(k)&&!k,s=(q?d:c).defer(),t=s.promise,n;n=a.defer(function()
     {try{s.resolve(f.apply(null,m))}catch(a){s.reject(a),e(a)}finally{delete
     g[t.$$timeou...
146  b.$apply()},l);t.$$timeoutId=n;g[n]=s;return t}var g={};f.cancel=function(b){return
     b&&b.$$timeoutId in g?(g[b.$$timeoutId].reject("canceled"),delete
     g[b.$$timeoutId],a.defer.cancel(b.$$timeoutId)):!1};return f}]}function
     Aa(b){Va&&(Z.setAttribute("href",b),b=Z.href);Z.setAttribute("href",b);return{href:Z.h
     ref,protocol:Z.protocol?Z.protocol.replace(/:$/,""):"",host:Z.host,search:Z.search?Z.s
     earch.replace(/^\?/,""):"",hash:Z.hash?Z.hash.replace(/^#/,""):"",hostname:Z.hostname,
     port:Z.port,pathname...
```

```
147                 Z.pathname.charAt(0)?Z.pathname:"/"+Z.pathname}}function gd(b){b=H(b)?Aa(b):b;return
                    b.protocol===wd.protocol&&b.host===wd.host}function uf(){this.$get=qa(N)}function
                    xd(b){function a(a){try{return decodeURIComponent(a)}catch(b){return a}}var
                    c=b[0]||{},d={},e="";return function(){var
                    b,g,h,l,k;b=c.cookie||"";if(b!==e)for(e=b,b=e.split(";
                    "),d={},h=0;h<b.length;h++)g=b[h],l=g.indexOf("="),0<l&&(k=a(g.substring(0,l)),d[k]===
                    u&&(d[k]=a(g.substring(l+1))));return d}}function yf(){this.$get=xd}funct...
148                 d){if(D(c)){var e={};n(c,function(b,c){e[c]=a(c,b)});return e}return
                    b.factory(c+"Filter",d)}this.register=a;this.$get=["$injector",function(a){return
                    function(b){return
                    a.get(b+"Filter")}}];a("currency",yd);a("date",zd);a("filter",$f);a("json",ag);a("limi
                    tTo",bg);a("lowercase",cg);a("number",Ad);a("orderBy",Bd);a("uppercase",dg)}function
                    $f(){return function(b,a,c){if(!Da(b)){if(null==b)return b;throw
                    G("filter")("notarray",b);}var d;switch(gc(a)){case "function":break;case
                    "boolean":case "null...
```

Category: Header Manipulation (3 Issues)



Abstract:

The method processRequest() in FileServlet.java includes unvalidated data in an HTTP response header on line 159. This enables attacks such as cache-poisoning, cross-site scripting, cross-user defacement, page hijacking, cookie manipulation or open redirect.

Explanation:

Header Manipulation vulnerabilities occur when:

1. Data enters a web application through an untrusted source, most frequently an HTTP request.

2. The data is included in an HTTP response header sent to a web user without being validated.

As with many software security vulnerabilities, Header Manipulation is a means to an end, not an end in itself. At its root, the vulnerability is straightforward: an attacker passes malicious data to a vulnerable application, and the application includes the data in an HTTP response header.

One of the most common Header Manipulation attacks is HTTP Response Splitting. To mount a successful HTTP Response Splitting exploit, the application must allow input that contains CR (carriage return, also given by %0d or \r) and LF (line feed, also given by %0a or \n)characters into the header. These characters not only give attackers control of the remaining headers and body of the response the application intends to send, but also allows them to create additional responses entirely under their control.

Many of today's modern application servers will prevent the injection of malicious characters into HTTP headers. For example, recent versions of Apache Tomcat will throw an IllegalArgumentException if you attempt to set a header with prohibited characters. If your application server prevents setting headers with new line characters, then your application is not vulnerable to HTTP Response Splitting. However, solely filtering for new line characters can leave an application vulnerable to Cookie Manipulation or Open Redirects, so care must still be taken when setting HTTP headers with user input.

Example: The following code segment reads the name of the author of a weblog entry, author, from an HTTP request and sets it in a cookie header of an HTTP response.

```
String author = request.getParameter(AUTHOR_PARAM);
...
Cookie cookie = new Cookie("author", author);
cookie.setMaxAge(cookieExpiration);
response.addCookie(cookie);
```

Assuming a string consisting of standard alpha-numeric characters, such as "Jane Smith", is submitted in the request the HTTP response including this cookie might take the following form:

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Jane Smith
...
```

However, because the value of the cookie is formed of unvalidated user input the response will only maintain this form if the value submitted for AUTHOR_PARAM does not contain any CR and LF characters. If an attacker submits a malicious string, such as "Wiley Hacker\r\nHTTP/1.1 200 OK\r\n...", then the HTTP response would be split into two responses of the following form:

```
HTTP/1.1 200 OK
```

...
Set-Cookie: author=Wiley Hacker

HTTP/1.1 200 OK

...

Clearly, the second response is completely controlled by the attacker and can be constructed with any header and body content desired. The ability of attacker to construct arbitrary HTTP responses permits a variety of resulting attacks, including: cross-user defacement, web and browser cache poisoning, cross-site scripting, and page hijacking.

Cross-User Defacement: An attacker will be able to make a single request to a vulnerable server that will cause the server to create two responses, the second of which may be misinterpreted as a response to a different request, possibly one made by another user sharing the same TCP connection with the server. This can be accomplished by convincing the user to submit the malicious request themselves, or remotely in situations where the attacker and the user share a common TCP connection to the server, such as a shared proxy server. In the best case, an attacker may leverage this ability to convince users that the application has been hacked, causing users to lose confidence in the security of the application. In the worst case, an attacker may provide specially crafted content designed to mimic the behavior of the application but redirect private information, such as account numbers and passwords, back to the attacker.

Cache Poisoning: The impact of a maliciously constructed response can be magnified if it is cached either by a web cache used by multiple users or even the browser cache of a single user. If a response is cached in a shared web cache, such as those commonly found in proxy servers, then all users of that cache will continue receive the malicious content until the cache entry is purged. Similarly, if the response is cached in the browser of an individual user, then that user will continue to receive the malicious content until the cache entry is purged, although only the user of the local browser instance will be affected.

Cross-Site Scripting: Once attackers have control of the responses sent by an application, they have a choice of a variety of malicious content to provide users. Cross-site scripting is common form of attack where malicious JavaScript or other code included in a response is executed in the user's browser. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data like cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site. The most common and dangerous attack vector against users of a vulnerable application uses JavaScript to transmit session and authentication information back to the attacker who can then take complete control of the victim's account.

Page Hijacking: In addition to using a vulnerable application to send malicious content to a user, the same root vulnerability can also be leveraged to redirect sensitive content generated by the server and intended for the user to the attacker instead. By submitting a request that results in two responses, the intended response from the server and the response generated by the attacker, an attacker may cause an intermediate node, such as a shared proxy server, to misdirect a response generated by the server for the user to the attacker. Because the request made by the attacker generates two responses, the first is interpreted as a response to the attacker's request, while the second remains in limbo. When the user makes a legitimate request through the same TCP connection, the attacker's request is already waiting and is interpreted as a response to the victim's request. The attacker then sends a second request to the server, to which the proxy server responds with the server generated request intended for the victim, thereby compromising any sensitive information in the headers or body of the response intended for the victim.

Cookie Manipulation: When combined with attacks like Cross-Site Request Forgery, attackers may change, add to, or even overwrite a legitimate user's cookies.

Open Redirect: Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.

Recommendations:

The solution to Header Manipulation is to ensure that input validation occurs in the correct places and checks for the correct properties.

Since Header Manipulation vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating responses dynamically, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for Header Manipulation.

Web applications must validate their input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for Header Manipulation is generally relatively easy. Despite its value, input validation for Header Manipulation does not take the place of rigorous output validation. An application may accept input through a shared data store or other trusted source, and that data store may accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means the best way to prevent Header Manipulation vulnerabilities is to validate everything that enters the application or leaves the application destined for the user.

The most secure approach to validation for Header Manipulation is to create a whitelist of safe characters that are allowed to appear in HTTP response headers and accept input composed exclusively of characters in the approved set. For example, a valid name might only include alpha-numeric characters or an account number might only include digits 0-9.

A more flexible, but less secure approach is known as blacklisting, which selectively rejects or escapes potentially dangerous characters before using the input. In order to form such a list, you first need to understand the set of characters that hold special meaning in HTTP response headers. Although the CR and LF characters are at the heart of an HTTP response splitting attack, other characters, such as ':' (colon) and '=' (equal), have special meaning in response headers as well.

After you identify the correct points in an application to perform validation for Header Manipulation attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. The application should reject any input destined to be included in HTTP response headers that contains special characters, particularly CR and LF, as invalid.

Many application servers attempt to limit an application's exposure to HTTP response splitting vulnerabilities by providing implementations for the functions responsible for setting HTTP headers and cookies that perform validation for the characters essential to an HTTP response splitting attack. Do not rely on the server running your application to make it secure. When an application is developed there are no guarantees about what application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will also stay in sync.

Tips:

1. Many HttpServletRequest implementations return a URL-encoded string from getHeader(), will not cause a HTTP response splitting issue unless it is decoded first because the CR and LF characters will not carry a meta-meaning in their encoded form. However, this behavior is not specified in the J2EE standard and varies by implementation. Furthermore, even encoded user input returned from getHeader() can lead to other vulnerabilities, including open redirects and other HTTP header tampering.

2. A number of modern web frameworks provide mechanisms to perform user input validation (including Struts and Spring MVC). To highlight the unvalidated sources of input, the Fortify Secure Coding Rulepacks dynamically re-prioritize the issues reported by Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. We refer to this feature as Context-Sensitive Ranking. To further assist the Fortify user with the auditing process, the Fortify Software Security Research group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

3. Fortify RTA adds protection against this category.

FileServlet.java, line 159 (Header Manipulation)

Fortify Priority:	High	**Folder**	High
Kingdom:	Input Validation and Representation		
Abstract:	The method processRequest() in FileServlet.java includes unvalidated data in an HTTP response header on line 159. This enables attacks such as cache-poisoning, cross-site scripting, cross-user defacement, page hijacking, cookie manipulation or open redirect.		

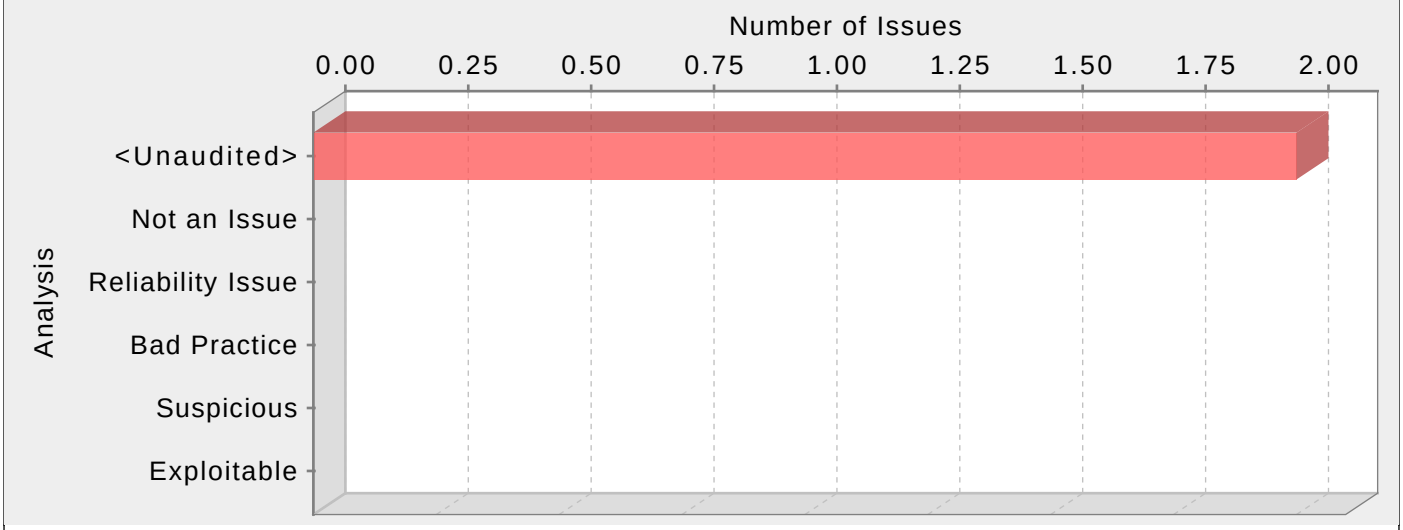
Source: FileServlet.java:126 javax.servlet.http.HttpServletRequest.getPathInfo()

```
124
125   // Get requested file by path info.
126   String requestedFile = request.getPathInfo().replace("/owa/fileServlet", "");
127
128   // Check if file is actually supplied to the request URL.
```

Sink: FileServlet.java:159 javax.servlet.http.HttpServletResponse.setHeader()

```
157   if (ifNoneMatch != null && matches(ifNoneMatch, eTag)) {
158    response.setStatus(HttpServletResponse.SC_NOT_MODIFIED);
159    response.setHeader("ETag", eTag); // Required in 304.
160    response.setDateHeader("Expires", expires); // Postpone cache with 1 week.
161    return;
```

Category: File Disclosure: J2EE (2 Issues)



Abstract:

On line 55 of OwaFilter.java, the method doFilter() invokes a server side forward using a path built with unvalidated input. This could allow an attacker to download application binaries or view arbitrary files within protected directories.

Explanation:

A file disclosure occur when:

1. Data enters a program from an untrusted source.

2. The data is used to dynamically construct a path.

Example 1: The following code takes untrusted data and uses it to build a path which is used in a server side forward.

```
...
String returnURL = request.getParameter("returnURL");
RequestDispatcher rd = request.getRequestDispatcher(returnURL);
rd.forward();
...
```

Example 2: The following code takes untrusted data and uses it to build a path which is used in a server side forward.

```
...
<% String returnURL = request.getParameter("returnURL"); %>
<jsp:include page="<%=returnURL%>" />
...
```

If an attacker provided a URL with the request parameter matching a sensitive file location, they would be able to view that file. For example, "http://www.yourcorp.com/webApp/logic?returnURL=WEB-INF/applicationContext.xml" would allow them to view the applicationContext.xml of the application.

After the attacker had the applicationContext.xml, they could locate and download other configuration files referenced in the applicationContext.xml or even class or jar files. This would allow attackers to gain sensitive infomation about an application and target it for other types of attack.

Recommendations:

Do not use untrusted data to direct requests to server-side resources. Instead, use a level of indirection between locations and paths.

Instead of the following:

```
< a href="http://www.yourcorp.com/webApp/logic?nextPage=WEB-INF/signup.jsp">New Customer</a>
```

Use the following:

```
< a href="http://www.yourcorp.com/webApp/logic?nextPage=newCustomer">New Customer</a>
```

The server-side logic would have a map keyed by logical names to server-side paths such that the path stored under the key "newCustomer" would be "/WEB-INF/signup.jsp".

OwaFilter.java, line 55 (File Disclosure: J2EE)

Fortify Priority:	High	**Folder**	High

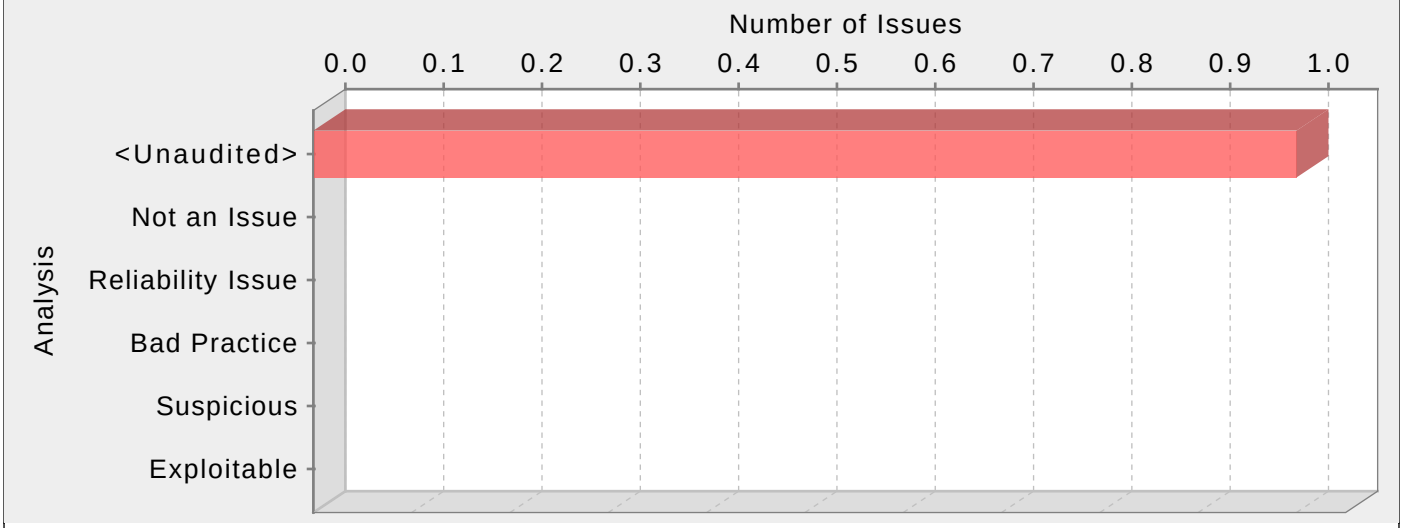
Kingdom:	API Abuse
Abstract:	On line 55 of OwaFilter.java, the method doFilter() invokes a server side forward using a path built with unvalidated input. This could allow an attacker to download application binaries or view arbitrary files within protected directories.
Source:	OwaFilter.java:47 javax.servlet.http.HttpServletRequest.getRequestURL()

```
45    String requestURL = null;
46    if (isFullBasePath(owaBasePath)) {
47     requestURL = request.getRequestURL().toString();
48    } else {
49     requestURL = request.getServletPath();
```

Sink:	OwaFilter.java:55 javax.servlet.ServletRequest.getRequestDispatcher()

```
53     if (requestURL.startsWith(owaBasePath)) {
54      String newURL = requestURL.replace(owaBasePath, "/ms/owa/fileServlet");
55      req.getRequestDispatcher(newURL).forward(req, res);
56     } else {
57      chain.doFilter(req, res);
```

Category: Open Redirect (1 Issues)



Abstract:

The file RedirectServlet.java passes unvalidated data to an HTTP redirect function on line 57. Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.

Explanation:

Redirects allow web applications to direct users to different pages within the same application or to external sites. Applications utilize redirects to aid in site navigation and, in some cases, to track how users exit the site. Open redirect vulnerabilities occur when a web application redirects clients to any arbitrary URL that can be controlled by an attacker.

Attackers may utilize open redirects to trick users into visiting a URL to a trusted site and redirecting them to a malicious site. By encoding the URL, an attacker is able to make it more difficult for end-users to notice the malicious destination of the redirect, even when it is passed as a URL parameter to the trusted site. Open redirects are often abused as part of phishing scams to harvest sensitive end-user data.

Example 1: The following JSP code instructs the user's browser to open a URL parsed from the dest request parameter when a user clicks the link.

```
<%
...
String strDest = request.getParameter("dest");
pageContext.forward(strDest);
...
%>
```

If a victim received an email instructing them to follow a link to "http://trusted.example.com/ecommerce/redirect.asp?dest=www.wilyhacker.com", the user would likely click on the link believing they would be transferred to the trusted site. However, when the victim clicks the link, the code in Example 1 will redirect the browser to "http://www.wilyhacker.com".

Many users have been educated to always inspect URLs they receive in emails to make sure the link specifies a trusted site they know. However, if the attacker Hex encoded the destination url as follows:

"http://trusted.example.com/ecommerce/redirect.asp?dest=%77%69%6C%79%68%61%63%6B%65%72%2E%63%6F%6D"

then even a savvy end-user may be fooled into following the link.

Recommendations:

Unvalidated user input should not be allowed to control the destination URL in a redirect. Instead, use a level of indirection: create a list of legitimate URLs that users are allowed to specify and only allow users to select from the list. With this approach, input provided by users is never used directly to specify a URL for redirects.

Example 2: The following code references an array populated with valid URLs. The link the user clicks passes in the array index that corresponds to the desired URL.

```
<%
...
try {
int strDest = Integer.parseInt(request.getParameter("dest"));
if((strDest >= 0) && (strDest <= strURLArray.length -1 ))
{
```

```
strFinalURL = strURLArray[strDest];
pageContext.forward(strFinalURL);
}
}
catch (NumberFormatException nfe) {
// Handle exception
...
}
...
%>
```

In some situations this approach is impractical because the set of legitimate URLs is too large or too hard to keep track of. In such cases, use a similar approach to restrict the domains that users can be redirected to, which can at least prevent attackers from sending users to malicious external sites.

Tips:

1. A number of modern web frameworks provide mechanisms to perform user input validation (including Struts and Spring MVC). To highlight the unvalidated sources of input, the Fortify Secure Coding Rulepacks dynamically re-prioritize the issues reported by Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. We refer to this feature as Context-Sensitive Ranking. To further assist the Fortify user with the auditing process, the Fortify Software Security Research group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

2. Fortify RTA adds protection against this category.

RedirectServlet.java, line 57 (Open Redirect)

Fortify Priority:	Critical	**Folder**	Critical
Kingdom:	Input Validation and Representation		
Abstract:	The file RedirectServlet.java passes unvalidated data to an HTTP redirect function on line 57. Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.		

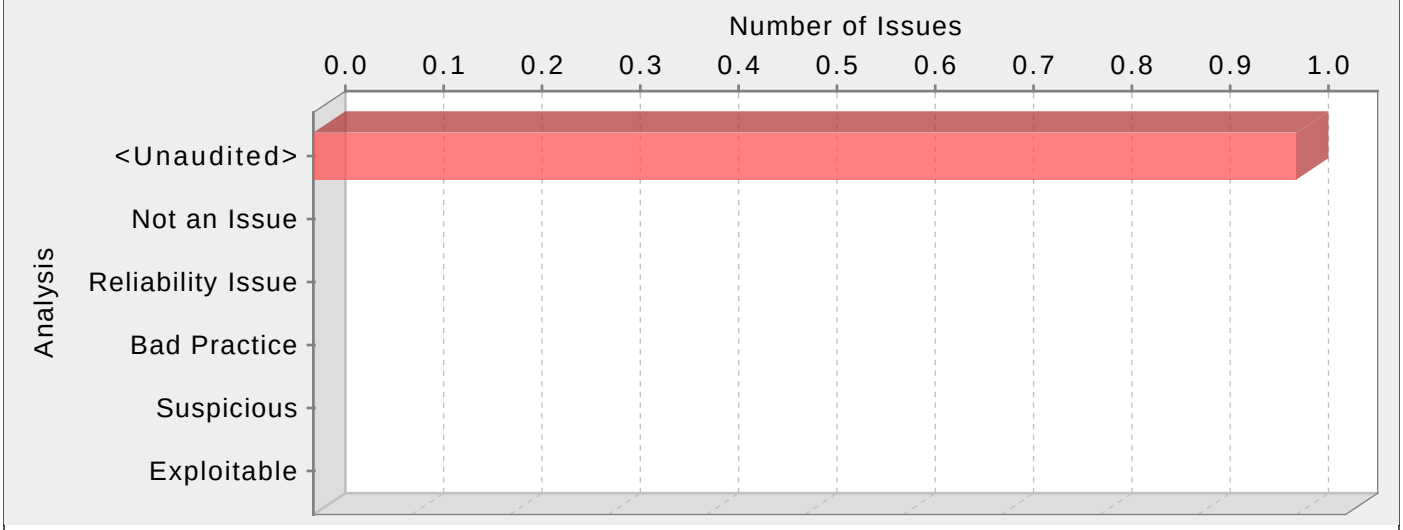
Source: RedirectServlet.java:54 javax.servlet.http.HttpServletRequest.getRequestURL()

```
52    private void processRequest(HttpServletRequest request, HttpServletResponse response,
      boolean content)
53            throws IOException {
54      String url = request.getRequestURL().toString().replace("/ms/owa/redirectServlet",
      "/owa");
55      //@TODO redirecting to original url after login in openmrs.
56      String loginUrl = Context.getAdministrationService().getGlobalProperty("login.url",
      "login.htm");
```

Sink: RedirectServlet.java:57 javax.servlet.http.HttpServletResponse.sendRedirect()

```
55      //@TODO redirecting to original url after login in openmrs.
56      String loginUrl = Context.getAdministrationService().getGlobalProperty("login.url",
      "login.htm");
57      response.sendRedirect(request.getContextPath() + "/" + loginUrl + "?redirect=" + url);
58    }
59  }
```

Category: Path Manipulation (1 Issues)



Abstract:

Attackers are able to control the file system path argument to File() at FileServlet.java line 137, which allows them to access or modify otherwise protected files.

Explanation:

Path manipulation errors occur when the following two conditions are met:

1. An attacker is able to specify a path used in an operation on the file system.

2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted.

For example, the program may give the attacker the ability to overwrite the specified file or run with a configuration controlled by the attacker.

Example 1: The following code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "../../tomcat/conf/server.xml", which causes the application to delete one of its own configuration files.

```
String rName = request.getParameter("reportName");
File rFile = new File("/usr/local/apfr/reports/" + rName);
...
rFile.delete();
```

Example 2: The following code uses input from a configuration file to determine which file to open and echo back to the user. If the program runs with adequate privileges and malicious users can change the configuration file, they can use the program to read any file on the system that ends with the extension .txt.

```
fis = new FileInputStream(cfg.getProperty("sub")+".txt");
amt = fis.read(arr);
out.println(arr);
```

Some think that in the mobile world, classic vulnerabilities, such as path manipulation, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 3: The following code adapts Example 1 to the Android platform.

```
...
String rName = this.getIntent().getExtras().getString("reportName");
File rFile = getBaseContext().getFileStreamPath(rName);
...
rFile.delete();
...
```

Recommendations:

The best way to prevent path manipulation is with a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name.

In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to keep track of. Programmers often resort to blacklisting in these situations. Blacklisting selectively rejects or escapes potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a whitelist of characters that are allowed to appear in the resource name and accept input composed exclusively of characters in the approved set.

Tips:

1. If the program is performing custom input validation you are satisfied with, use the Fortify Custom Rules Editor to create a cleanse rule for the validation routine.

2. Implementation of an effective blacklist is notoriously difficult. One should be skeptical if validation logic requires blacklisting. Consider different types of input encoding and different sets of meta-characters that might have special meaning when interpreted by different operating systems, databases, or other resources. Determine whether or not the blacklist can be updated easily, correctly, and completely if these requirements ever change.

3. A number of modern web frameworks provide mechanisms to perform user input validation (including Struts and Spring MVC). To highlight the unvalidated sources of input, the Fortify Secure Coding Rulepacks dynamically re-prioritize the issues reported by Fortify Static Code Analyzer by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. We refer to this feature as Context-Sensitive Ranking. To further assist the Fortify user with the auditing process, the Fortify Software Security Research group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

FileServlet.java, line 137 (Path Manipulation)

Fortify Priority:	Critical	**Folder**	Critical
Kingdom:	Input Validation and Representation		
Abstract:	Attackers are able to control the file system path argument to File() at FileServlet.java line 137, which allows them to access or modify otherwise protected files.		

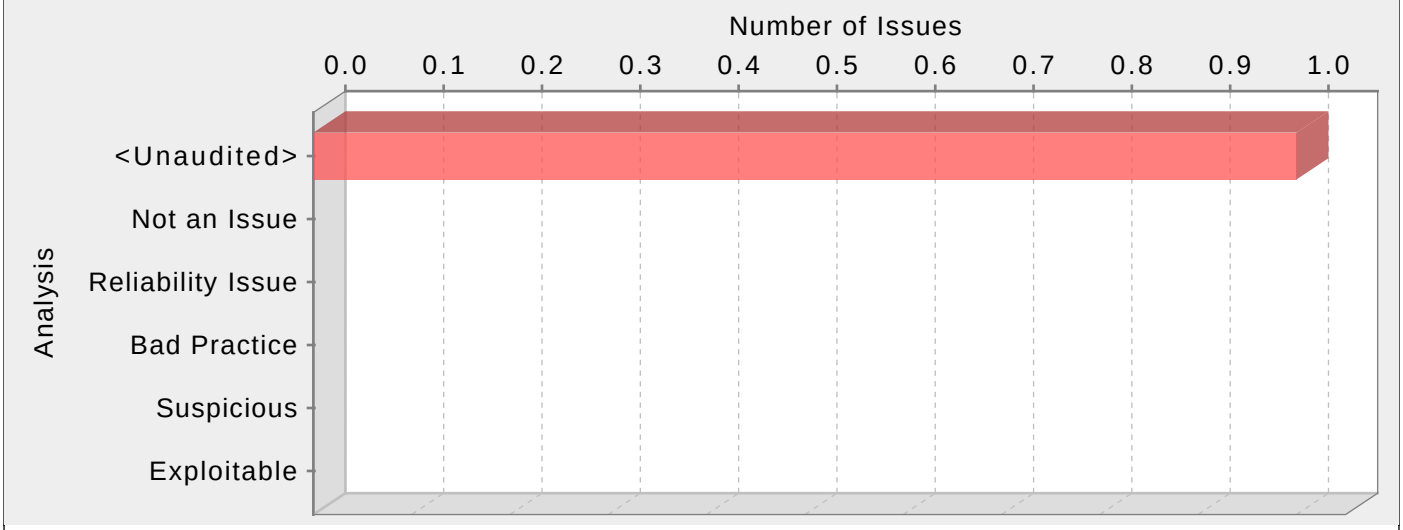
Source: FileServlet.java:126 javax.servlet.http.HttpServletRequest.getPathInfo()

```
124
125                 // Get requested file by path info.
126                 String requestedFile = request.getPathInfo().replace("/owa/fileServlet", "");
127
128                 // Check if file is actually supplied to the request URL.
```

Sink: FileServlet.java:137 java.io.File.File()

```
135
136                 // URL-decode the file name (might contain spaces and on) and prepare file object.
137                 File file = new File(basePath, URLDecoder.decode(requestedFile, "UTF-8"));
138
139                 // Check if file actually exists in filesystem.
```

Category: Race Condition: Singleton Member Field (1 Issues)



Abstract:

The class DefaultAppManager is a singleton, so the member field apps is shared between users. The result is that one user could see another user's data.

Explanation:

Many Servlet developers do not understand that a Servlet is a singleton. There is only one instance of the Servlet, and that single instance is used and re-used to handle multiple requests that are processed simultaneously by different threads.

A common result of this misunderstanding is that developers use Servlet member fields in such a way that one user may inadvertently see another user's data. In other words, storing user data in Servlet member fields introduces a data access race condition.

Example 1: The following Servlet stores the value of a request parameter in a member field and then later echoes the parameter value to the response output stream.

```
public class GuestBook extends HttpServlet {

String name;

protected void doPost (HttpServletRequest req, HttpServletResponse res) {
name = req.getParameter("name");
...
out.println(name + ", thanks for visiting!");
}
}
```

While this code will work perfectly in a single-user environment, if two users access the Servlet at approximately the same time, it is possible for the two request handler threads to interleave in the following way:

Thread 1: assign "Dick" to name
Thread 2: assign "Jane" to name
Thread 1: print "Jane, thanks for visiting!"
Thread 2: print "Jane, thanks for visiting!"

Thereby showing the first user the second user's name.

Recommendations:

Do not use Servlet member fields for anything but constants. (i.e. make all member fields static final).

Developers are often tempted to use Servlet member fields for user data when they need to transport data from one region of code to another. If this is your aim, consider declaring a separate class and using the Servlet only to "wrap" this new class.

Example 2: The bug in Example 1 can be corrected in the following way:

```
public class GuestBook extends HttpServlet {

protected void doPost (HttpServletRequest req, HttpServletResponse res) {
GBRequestHandler handler = new GBRequestHandler();
handler.handle(req, res);
}
```

```
}
public class GBRequestHandler {
String name;
public void handle(HttpServletRequest req, HttpServletResponse res) {
name = req.getParameter("name");
...
out.println(name + ", thanks for visiting!");
}
}
```

Alternatively, a Servlet can utilize synchronized blocks to access servlet instance variables but using synchronized blocks may cause significant performance problems.

Please notice that wrapping the field access within a synchronized block will only prevent the issue if all read and write operations on that member are performed within the same synchronized block or method.

Example 3: Wrapping the Example 1 write operation (assignment) in a synchronized block will not fix the problem since the threads will have to get a lock to modify name field, but they will release the lock afterwards, allowing a second thread to change the value again. If, after changing the name value, the first thread resumes execution, the value printed will be the one assigned by the second thread:

```
public class GuestBook extends HttpServlet {
String name;
protected void doPost (HttpServletRequest req, HttpServletResponse res) {
synchronized(name) {
name = req.getParameter("name");
}
...
out.println(name + ", thanks for visiting!");
}
}
```

In order to fix the race condition, all the write and read operations on the shared member field should be run atomically within the same synchronized block:

```
public class GuestBook extends HttpServlet {
String name;
protected void doPost (HttpServletRequest req, HttpServletResponse res) {
synchronized(name) {
name = req.getParameter("name");
...
out.println(name + ", thanks for visiting!");
}
}
}
```

DefaultAppManager.java, line 299 (Race Condition: Singleton Member Field)

Fortify Priority:	High	**Folder**	High
Kingdom:	Time and State		
Abstract:	The class DefaultAppManager is a singleton, so the member field apps is shared between users. The result is that one user could see another user's data.		

Sink: DefaultAppManager.java:299 AssignmentStatement()

```
297    }
298
299    this.apps = appList;
300    log.info("Detected apps: " + apps);
301    if (owaListeners != null) {
```

Detailed Project Summary

Files Scanned

Code base location: /srv/openmrs_code/org/openmrs/module/owa

Files Scanned:

.travis.yml yaml Dec 13, 2019 12:57:08 PM

/home/pgupta25/.fortify/sca19.1/build/owa/extracted/angularjs/sca.frontend.angularjs.aux.js secondary 5 Lines Jan 22, 2020 6:10:44 PM

/home/pgupta25/.fortify/sca19.1/build/owa/extracted/javascript/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/manage.html.js secondary Jan 22, 2020 6:10:44 PM

/home/pgupta25/.fortify/sca19.1/build/owa/extracted/javascript/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/settings.html.js secondary Jan 22, 2020 6:10:44 PM

/home/pgupta25/.fortify/sca19.1/build/owa/extracted/javascript/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/manage.html.js secondary Jan 22, 2020 6:10:44 PM

/home/pgupta25/.fortify/sca19.1/build/owa/extracted/javascript/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/settings.html.js secondary Jan 22, 2020 6:10:44 PM

/home/pgupta25/.fortify/sca19.1/build/owa/extracted/javascript/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/manage.html.js secondary Jan 22, 2020 6:10:44 PM

/home/pgupta25/.fortify/sca19.1/build/owa/extracted/javascript/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/settings.html.js secondary Jan 22, 2020 6:10:44 PM

OpenMRSFormatter.xml xml 27.9 KB Dec 13, 2019 12:57:08 PM

api/pom.xml xml 2.6 KB Dec 13, 2019 12:57:08 PM

api/src/main/java/org/openmrs/module/owa/App.java java 41 Lines 5.7 KB Dec 17, 2019 7:06:51 PM

api/src/main/java/org/openmrs/module/owa/AppActivities.java java 4 Lines 2.1 KB Dec 17, 2019 7:06:49 PM

api/src/main/java/org/openmrs/module/owa/AppDeveloper.java java 10 Lines 2.8 KB Dec 17, 2019 7:06:48 PM

api/src/main/java/org/openmrs/module/owa/AppIcons.java java 8 Lines 2.5 KB Dec 17, 2019 7:06:50 PM

api/src/main/java/org/openmrs/module/owa/AppManager.java java 5 Lines 3.8 KB Dec 17, 2019 7:06:49 PM

api/src/main/java/org/openmrs/module/owa/AppOpenmrs.java java 4 Lines 2.1 KB Dec 17, 2019 7:06:51 PM

api/src/main/java/org/openmrs/module/owa/OwaListener.java java Dec 17, 2019 7:06:48 PM

api/src/main/resources/messages.properties java_properties 1.6 KB Dec 13, 2019 12:57:08 PM

api/src/main/resources/messages_es.properties java_properties Dec 13, 2019 12:57:08 PM

api/src/main/resources/messages_fr.properties java_properties Dec 13, 2019 12:57:08 PM

api/src/main/resources/moduleApplicationContext.xml xml 1.6 KB Dec 13, 2019 12:57:08 PM

api/target/classes/messages.properties java_properties 1.1 KB Dec 18, 2019 12:01:29 PM

api/target/classes/messages_es.properties java_properties Dec 18, 2019 12:01:29 PM

api/target/classes/messages_fr.properties java_properties Dec 18, 2019 12:01:29 PM

api/target/classes/moduleApplicationContext.xml xml 1.6 KB Dec 18, 2019 12:01:29 PM

api/target/maven-archiver/pom.properties java_properties Dec 18, 2019 12:01:33 PM

api/target/maven-java-formatter-cache.properties java_properties Dec 18, 2019 12:01:29 PM

omod/pom.xml xml 8.3 KB Dec 13, 2019 12:57:08 PM

omod/src/main/java/org/openmrs/module/owa/activator/OwaActivator.java java 40 Lines 5.3 KB Dec 17, 2019 7:09:22 PM

omod/src/main/java/org/openmrs/module/owa/extension/html/AdminList.java java 7 Lines 1.2 KB Dec 17, 2019 7:09:39 PM

omod/src/main/java/org/openmrs/module/owa/filter/OwaFilter.java java 19 Lines 2.5 KB Dec 13, 2019 12:57:08 PM

omod/src/main/java/org/openmrs/module/owa/impl/DefaultAppManager.java java 88 Lines 9.9 KB Dec 17, 2019 7:09:30 PM

omod/src/main/java/org/openmrs/module/owa/servlet/FileServlet.java java 142 Lines 17.9 KB Dec 17, 2019 7:09:33 PM

omod/src/main/java/org/openmrs/module/owa/servlet/RedirectServlet.java java 6 Lines 2.1 KB Dec 17, 2019 7:09:31 PM

omod/src/main/java/org/openmrs/module/owa/utils/OwaUtils.java java 10 Lines 1.5 KB Dec 17, 2019 7:09:26 PM

omod/src/main/java/org/openmrs/module/owa/web/controller/AddAppController.java java 28 Lines 5.5 KB Dec 17, 2019 7:09:39 PM

omod/src/main/java/org/openmrs/module/owa/web/controller/InstallAppRequestObject.java java 4 Lines Dec 17, 2019 7:09:37 PM

omod/src/main/java/org/openmrs/module/owa/web/controller/OwaManageController.java java 29 Lines 4 KB Dec 17, 2019 7:09:34

PM
omod/src/main/java/org/openmrs/module/owa/web/controller/OwaManagerRestController.java java 3 Lines Dec 17, 2019 7:09:35 PM
omod/src/main/java/org/openmrs/module/owa/web/controller/OwaRestController.java java 103 Lines 9.5 KB Dec 17, 2019 7:09:34 PM
omod/src/main/java/org/openmrs/module/owa/web/controller/SettingsFormController.java java 25 Lines 4.8 KB Dec 17, 2019 7:09:34 PM
omod/src/main/resources/config.xml xml 3.2 KB Dec 13, 2019 12:57:08 PM
omod/src/main/resources/webModuleApplicationContext.xml xml 1.8 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/manage.html html 9 Lines 4.5 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/manage.jsp jsp 24 Lines 6.6 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/resources/javascript/angular-1.4.5.min.js typescript 287 Lines 143.4 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/resources/javascript/bootstrap-filestyle.min.js typescript 1 Lines 7 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/resources/javascript/deleteApp.js typescript 4 Lines Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/resources/javascript/jquery.dataTables.min.js typescript 160 Lines 79.8 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/resources/javascript/jquery.form.js typescript 525 Lines 40.1 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/resources/javascript/uploadApp.js typescript 27 Lines 1.6 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/settings.html html 1 Lines 2.2 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/settings.jsp jsp 7 Lines 2.9 KB Dec 13, 2019 12:57:08 PM
omod/src/main/webapp/template/localHeader.jsp jsp 6 Lines Dec 13, 2019 12:57:08 PM
omod/src/test/java/org/openmrs/module/owa/filter/OwaFilterTest.java java 27 Lines 4 KB Dec 17, 2019 7:09:42 PM
omod/src/test/java/org/openmrs/module/owa/impl/DefaultAppManagerTest.java java 37 Lines 5.2 KB Dec 17, 2019 7:09:44 PM
omod/src/test/java/org/openmrs/module/owa/utils/OwaUtilsTest.java java 7 Lines Dec 17, 2019 7:09:41 PM
omod/src/test/java/org/openmrs/module/owa/web/controller/AddAppControllerTest.java java 32 Lines 4.6 KB Dec 13, 2019 12:57:08 PM
omod/src/test/java/org/openmrs/module/owa/web/controller/OwaRestControllerTest.java java 76 Lines 9.3 KB Dec 17, 2019 7:09:45 PM
omod/target/classes/META-INF/maven/org.openmrs.module/owa-api/pom.properties java_properties Dec 18, 2019 12:01:34 PM
omod/target/classes/META-INF/maven/org.openmrs.module/owa-api/pom.xml xml 2.6 KB Dec 13, 2019 12:57:08 PM
omod/target/classes/config.xml xml 3 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/messages.properties java_properties 1.1 KB Dec 18, 2019 12:01:30 PM
omod/target/classes/messages_es.properties java_properties Dec 18, 2019 12:01:30 PM
omod/target/classes/messages_fr.properties java_properties Dec 18, 2019 12:01:30 PM
omod/target/classes/moduleApplicationContext.xml xml 1.6 KB Dec 18, 2019 12:01:30 PM
omod/target/classes/web/module/manage.html html 9 Lines 4.5 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/manage.jsp jsp 24 Lines 6.6 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/resources/javascript/angular-1.4.5.min.js typescript 287 Lines 143.4 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/resources/javascript/bootstrap-filestyle.min.js typescript 1 Lines 7 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/resources/javascript/deleteApp.js typescript 4 Lines Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/resources/javascript/jquery.dataTables.min.js typescript 160 Lines 79.8 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/resources/javascript/jquery.form.js typescript 525 Lines 40.1 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/resources/javascript/uploadApp.js typescript 27 Lines 1.6 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/settings.html html 1 Lines 2.2 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/settings.jsp jsp 7 Lines 2.9 KB Dec 18, 2019 12:01:43 PM
omod/target/classes/web/module/template/localHeader.jsp jsp 6 Lines Dec 18, 2019 12:01:43 PM
omod/target/classes/webModuleApplicationContext.xml xml 1.8 KB Dec 18, 2019 12:01:43 PM
omod/target/maven-archiver/pom.properties java_properties Dec 18, 2019 12:01:46 PM
omod/target/maven-java-formatter-cache.properties java_properties 1.9 KB Dec 18, 2019 12:01:41 PM
omod/target/owa-1.10.0/META-INF/maven/org.openmrs.module/owa-api/pom.properties java_properties Dec 18, 2019 12:01:46

PM
omod/target/owa-1.10.0/META-INF/maven/org.openmrs.module/owa-api/pom.xml xml 2.6 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/config.xml xml 3 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/messages.properties java_properties 1.1 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/messages_es.properties java_properties Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/messages_fr.properties java_properties Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/moduleApplicationContext.xml xml 1.6 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/manage.html html 9 Lines 4.5 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/manage.jsp jsp 24 Lines 6.6 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/resources/javascript/angular-1.4.5.min.js typescript 287 Lines 143.4 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/resources/javascript/bootstrap-filestyle.min.js typescript 1 Lines 7 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/resources/javascript/deleteApp.js typescript 4 Lines Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/resources/javascript/jquery.dataTables.min.js typescript 160 Lines 79.8 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/resources/javascript/jquery.form.js typescript 525 Lines 40.1 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/resources/javascript/uploadApp.js typescript 27 Lines 1.6 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/settings.html html 1 Lines 2.2 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/settings.jsp jsp 7 Lines 2.9 KB Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/web/module/template/localHeader.jsp jsp 6 Lines Dec 18, 2019 12:01:46 PM
omod/target/owa-1.10.0/webModuleApplicationContext.xml xml 1.8 KB Dec 18, 2019 12:01:46 PM
pom.xml xml 7.2 KB Dec 13, 2019 12:57:08 PM

Reference Elements

Classpath:

No classpath specified during translation

Libdirs:

No libdirs specified during translation

Rulepacks

Valid Rulepacks:

Name: Fortify Secure Coding Rules, Core, Java
Version: 2019.4.0.0009
ID: 06A6CC97-8C3F-4E73-9093-3E74C64A2AAF
SKU: RUL13003

Name: Fortify Secure Coding Rules, Core, Annotations
Version: 2019.4.0.0009
ID: 14EE50EB-FA1C-4AE8-8B59-39F952E21E3B
SKU: RUL13078

Name: Fortify Secure Coding Rules, Core, JavaScript
Version: 2019.4.0.0009
ID: BD292C4E-4216-4DB8-96C7-9B607BFD9584
SKU: RUL13059

Name: Fortify Secure Coding Rules, Core, Android
Version: 2019.4.0.0009
ID: FF9890E6-D119-4EE8-A591-83DCF4CA6952
SKU: RUL13093

Name: Fortify Secure Coding Rules, Extended, JavaScript
Version: 2019.4.0.0009
ID: C4D1969E-B734-47D3-87D4-73962C1D32E2
SKU: RUL13141

Name: Fortify Secure Coding Rules, Extended, Configuration
Version: 2019.4.0.0009
ID: CD6959FC-0C37-45BE-9637-BAA43C3A4D56
SKU: RUL13005

Name: Fortify Secure Coding Rules, Extended, Java
Version: 2019.4.0.0009
ID: AAAC0B10-79E7-4FE5-9921-F4903A79D317
SKU: RUL13007

Name: Fortify Secure Coding Rules, Extended, Content
Version: 2019.4.0.0009
ID: 9C48678C-09B6-474D-B86D-97EE94D38F17
SKU: RUL13067

Name: Fortify Secure Coding Rules, Core, Golang
Version: 2019.4.0.0009
ID: 1DCE79F8-AF6B-474D-A05A-5BFFC8B13DCD
SKU: RUL13218

Name: Fortify Secure Coding Rules, Extended, JSP
Version: 2019.4.0.0009
ID: 00403342-15D0-48C9-8E67-4B1CFBDEFCD2
SKU: RUL13026

External Metadata:
Version: 2019.4.0.0009

Name: CWE
ID: 3ADB9EE4-5761-4289-8BD3-CBFCC593EBBC
The Common Weakness Enumeration (CWE), co-sponsored and maintained by MITRE, is international in scope and free for public use. CWE provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design.

Name: CWE Top 25 2019
ID: 7AF935C9-15AA-45B2-8EEC-0EAE4194ACDE
The 2019 CWE Top 25 Most Dangerous Software Errors lists the most widespread and critical weaknesses that can lead to serious vulnerabilities in software (as demonstrated by the National Vulnerability Database). These weaknesses occur frequently, are often easy to find, and easy to exploit. They are dangerous because they will frequently enable attackers to completely take over

the software, steal data, or prevent the software from working at all. The list is the result of heuristic formula that the CWE Team used with a data-driven approach that leveraged the Common Vulnerabilities and Exposure (CVE), National Vulnerability Database (NVD), and Common Vulnerability Scoring System (CVSS). Due to the hiearchical nature of the CWE taxonomy, Fortify considers all CWE IDs which are children of a Top 25 entry, as included within the context of the entry due to the "CHILD-OF" relationship within the hierarchy. Exercise caution if using only this Top 25 list to prioritize auditing efforts because the software under analysis might not align with the assumptions of the heuristic used to define the Top 25. For example, many of these weaknesses are related to C-like langauges and the software under analysis might not be within the C-family of languages - thus, many CWEs would not be in scope.

Name: DISA CCI 2
ID: 7F037130-41E5-40F0-B653-7819A4B3E241
The purpose of a Defense Information Systems Agency (DISA) Control Correlation Identifier (CCI) is to provide a standard identifier for policy based requirements which connect high-level policy expressions and low-level technical implementations. Associated with each CCI is a description for each of the singular, actionable, statements compromising an information assurance (IA) control or IA best practice. Using CCI allows high-level policy framework security requirements to be decomposed and explicitly associated with low-level implementations, thus enabling the assessment of related compliance assessment results spanning heterogeneous technologies. The current IA controls and best practices associated with each CCI, that are specified in NIST SP 800-53 Revision 4, can be viewed using the DISA STIG Viewer.
The following table summarizes the number of issues identified across the different CCIs broken down by Fortify Priority Order. The status of a CCI is considered "In Place" when there are no issues reported for a given CCI.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, CCI-003187 is not considered "In Place". Similarly, if the project is missing a Micro Focus Fortify WebInspect scan, or the scan contains any critical findings, CCI-000366 and CCI-000256 are not considered "In Place".

Name: FISMA
ID: B40F9EE0-3824-4879-B9FE-7A789C89307C
The Federal Information Processing Standard (FIPS) 200 document is part of the official series of publications, issued by the National Institute of Standards and Technology (NIST), relating to standards and guidelines adopted and promulgated under the provisions of the Federal Information Security Management Act (FISMA). Specifically, FIPS Publication 200 specifies the "Minimum Security Requirements for Federal Information and Information Systems."

Name: GDPR
ID: 771C470C-9274-4580-8556-C12F5E4BEC51
The EU General Data Protection Regulation (GDPR) replaces the Data Protection Directive 95/46/EC and was designed to harmonize data privacy laws across Europe, to protect and empower all EU citizens data privacy and to reshape the way organizations across the region approach data privacy. Going into effect on May 25, 2018, GDPR provides a framework for organizations on how to handle personal data. According to GDPR regulation personal data "means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person." GDPR articles that pertain to application security and require businesses to protect personal data during design and development of its product and services are:

- Article 25, Data protection by design and by default - which requires "The controller shall implement appropriate technical and organisational measures for ensuring that, by default, only personal data which are necessary for each specific purpose of the processing are processed."
- Article 32, Security of processing - which requires businesses to protect its systems and applications "from accidental or unlawful destruction, loss, alteration, unauthorized disclosure of, or access to personal data". This report may be used by organizations as a framework to help identify and protect personal data as it relates to application security.

Name: MISRA C 2012
ID: 555A3A66-A0E1-47AF-910C-3F19A6FB2506
Now in its third edition, the Motor Industry Software Reliability Association (MISRA) C Guidelines describe a subset of the C programming language in which there is reduced risk of introducing mistakes in critical systems. While the MISRA C Guidelines focus upon safety-related software development, a subset of the rules also reflect security properties. Fortify interprets the MISRA C Guidelines under the context of security and provides correlation of security vulnerability categories to the rules defined by MISRA. Fortify provides these security focused detection mechanism with the standard rulepacks, however, further support of the MISRA C Guidelines related to safety can be added through the use of custom rules. The results in this report can assist in the creation of a compliance matrix for MISRA.

Name: MISRA C++ 2008
ID: 5D4B75A1-FC91-4B4B-BD4D-C81BBE9604FA
The Motor Industry Software Reliability Association (MISRA) C++ Guidelines describe a subset of the C++ programming language in which there is reduced risk of introducing mistakes in critical systems. While the MISRA C++ Guidelines focus upon safety-related software development, a subset of the rules also reflect security properties. Fortify interprets the MISRA C++ Guidelines under the context of security and provides correlation of security vulnerability categories to the rules defined by MISRA. Fortify provides these security focused detection mechanism with the standard rulepacks, however, further support of the MISRA C++ Guidelines related to safety can be added through the use of custom rules. The results in this report can assist in the creation of a compliance matrix for MISRA.

Name: NIST SP 800-53 Rev.4
ID: 1114583B-EA24-45BE-B7F8-B61201BACDD0
NIST Special Publication 800-53 Revision 4 provides a list of security and privacy controls designed to protect federal organizations and information systems from security threats. The following table summarizes the number of issues identified across the different controls and broken down by Fortify Priority Order.

Name: OWASP Mobile 2014
ID: EEE3F9E7-28D6-4456-8761-3DA56C36F4EE
The OWASP Mobile Top 10 Risks 2014 provides a powerful awareness document for mobile application security. The OWASP Mobile Top 10 represents a broad consensus about what the most critical mobile application security flaws are. Project members include a variety of security experts from around the world who have shared their expertise to produce this list.

Name: OWASP Top 10 2004
ID: 771C470C-9274-4580-8556-C023E4D3ADB4
The OWASP Top Ten 2004 provides a powerful awareness document for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are. Project members include a variety of security experts from around the world who have shared their expertise to produce this list.

Name: OWASP Top 10 2007
ID: 1EB1EC0E-74E6-49A0-BCE5-E6603802987A
The OWASP Top Ten 2007 provides a powerful awareness document for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are. Project members include a variety of security experts from around the world who have shared their expertise to produce this list.

Name: OWASP Top 10 2010
ID: FDCECA5E-C2A8-4BE8-BB26-76A8ECD0ED59
The OWASP Top Ten 2010 provides a powerful awareness document for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are. Project members include a variety of security experts from around the world who have shared their expertise to produce this list.

Name: OWASP Top 10 2013

ID: 1A2B4C7E-93B0-4502-878A-9BE40D2A25C4
The OWASP Top Ten 2013 provides a powerful awareness document for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are. Project members include a variety of security experts from around the world who have shared their expertise to produce this list.

Name: OWASP Top 10 2017
ID: 3C6ECB67-BBD9-4259-A8DB-B49328927248
The OWASP Top Ten 2017 provides a powerful awareness document for web application security focused on informing the community about the consequences of the most common and most important web application security weaknesses. The OWASP Top Ten represents a broad agreement about what the most critical web application security flaws are with consensus being drawn from data collection and survey results. Project members include a variety of security experts from around the world who have shared their expertise to produce this list.

Name: PCI 1.1
ID: CBDB9D4D-FC20-4C04-AD58-575901CAB531
The Payment Card Industry (PCI) Data Security Standard (DSS) 1.1 compliance standard describes 12 requirements which are organized into 6 logically related groups, which are "control objectives". PCI DSS requirements are applicable if Primary Account Number (PAN) is stored, processed, or transmitted by the system.

Name: PCI 1.2
ID: 57940BDB-99F0-48BF-BF2E-CFC42BA035E5
Payment Card Industry Data Security Standard Version 1.2 description

Name: PCI 2.0
ID: 8970556D-7F9F-4EA7-8033-9DF39D68FF3E
The PCI DSS 2.0 compliance standard, particularly sections 6.3, 6.5, and 6.6, references the OWASP Top 10 vulnerability categories as the core categories that must be tested for and remediated. The following table summarizes the number of issues identified across the different PCI DSS requirements and broken down by Fortify Priority Order.

Name: PCI 3.0
ID: E2FB0D38-0192-4F03-8E01-FE2A12680CA3
The following is a summary of the application security portions of Payment Card Industry (PCI) Data Security Standard (DSS) v3.0. Fortify tests for 32 application security related requirements across sections 1, 2, 3, 4, 6, 7, 8, and 10 of PCI DSS and reports whether each requirement is In Place or Not In Place to indicate whether requirements are satisfied or not. This report is intended to measure the level of adherence the specific application(s) possess when compared to PCI DSS 3.0 compliance and is not intended to serve as a comprehensive Report on Compliance (ROC). The information contained in this report is targeted at project managers, security auditors, and compliance auditors.

Name: PCI 3.1
ID: AC0D18CF-C1DA-47CF-9F1A-E8EC0A4A717E
The following is a summary of the application security portions of Payment Card Industry (PCI) Data Security Standard (DSS) v3.1. Fortify tests for 31 application security related requirements across sections 1, 2, 3, 4, 6, 7, 8, and 10 of PCI DSS and reports whether each requirement is In Place or Not In Place to indicate whether requirements are satisfied or not. This report is intended to measure the level of adherence the specific application(s) possess when compared to PCI DSS 3.1 compliance and is not intended to serve as a comprehensive Report on Compliance (ROC). The information contained in this report is targeted at project managers, security auditors, and compliance auditors.

Name: PCI 3.2
ID: 4E8431F9-1BA1-41A8-BDBD-087D5826751A
The following is a summary of the application security portions of Payment Card Industry (PCI) Data Security Standard (DSS) v3.2. Fortify tests for 31 application security related requirements across sections 1, 2, 3, 4, 6, 7, 8, and 10 of PCI DSS and reports

whether each requirement is In Place or Not In Place to indicate whether requirements are satisfied or not. This report is intended to measure the level of adherence the specific application(s) possess when compared to PCI DSS 3.2 compliance and is not intended to serve as a comprehensive Report on Compliance (ROC). The information contained in this report is targeted at project managers, security auditors, and compliance auditors.

Name: PCI 3.2.1
ID: EADE255F-6561-4EFE-AD31-2914F6BFA329
The following is a summary of the application security portions of Payment Card Industry (PCI) Data Security Standard (DSS) v3.2.1. Fortify tests for 31 application security related requirements across sections 1, 2, 3, 4, 6, 7, 8, and 10 of PCI DSS and reports whether each requirement is In Place or Not In Place to indicate whether requirements are satisfied or not. This report is intended to measure the level of adherence the specific application(s) possess when compared to PCI DSS 3.2.1 compliance and is not intended to serve as a comprehensive Report on Compliance (ROC). The information contained in this report is targeted at project managers, security auditors, and compliance auditors.

Name: PCI SSF 1.0
ID: 0F551543-AF0E-4334-BEDF-1DDCD5F4BF74
The following is a summary of the application security portions of the Secure Software Requirements and Assessment Procedures defined in the Payment Card Industry (PCI) Software Security Framework (SSF) v1.0. Fortify tests for 23 application security related control objectives across Control Objective sections 2, 3, 4, 5, 6, 7, 8, and A.2 of PCI SSF and reports whether each control objective is In Place or Not In Place to indicate whether requirements are satisfied or not. This report is intended to measure the level of adherence the specific application(s) possess when compared to PCI SSF 1.0 compliance and is not intended to serve as a comprehensive Report on Compliance (ROC). The information contained in this report is targeted at project managers, security auditors, and compliance auditors.

Name: SANS Top 25 2009
ID: 939EF193-507A-44E2-ABB7-C00B2168B6D8
The 2009 CWE/SANS Top 25 Programming Errors lists the most significant programming errors that can lead to serious software vulnerabilities. They occur frequently, are often easy to find, and easy to exploit. They are dangerous because they will frequently allow attackers to completely take over the software, steal data, or prevent the software from working at all. The list is the result of collaboration between the SANS Institute, MITRE, and many top software security experts.

Name: SANS Top 25 2010
ID: 72688795-4F7B-484C-88A6-D4757A6121CA
SANS Top 25 2010 Most Dangerous Software Errors provides an enumeration of the most widespread and critical errors, categorized by Common Weakness Enumeration (CWE) identifiers, that lead to serious vulnerabilities in software (http://cwe.mitre.org/). These software errors are often easy to find and exploit. The inherent danger in these errors is that they can allow an attacker to completely take over the software, steal data, or prevent the software from working at all.

Name: SANS Top 25 2011
ID: 92EB4481-1FD9-4165-8E16-F2DE6CB0BD63
SANS Top 25 2011 Most Dangerous Software Errors provides an enumeration of the most widespread and critical errors, categorized by Common Weakness Enumeration (CWE) identifiers, that lead to serious vulnerabilities in software (http://cwe.mitre.org/). These software errors are often easy to find and exploit. The inherent danger in these errors is that they can allow an attacker to completely take over the software, steal data, or prevent the software from working at all.

Name: STIG 3.1
ID: F2FA57EA-5AAA-4DDE-90A5-480BE65CE7E7
Security Technical Implementation Guide Version 3.1 description

Name: STIG 3.10
ID: 788A87FE-C9F9-4533-9095-0379A9B35B12

Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APP<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APP5080: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APP5100: CAT II is not considered "In Place".

Name: STIG 3.4
ID: 58E2C21D-C70F-4314-8994-B859E24CF855
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APP<I>ID</I>: CAT <I>SEV</I>]. DISA STIG identifies several severities with respect to vulnerabilities:

<LI>CAT I: allow an attacker immediate access into a machine, allow super user access, or bypass a firewall.</LI>
<LI>CAT II: provide information that have a high potential of giving access to an intruder.</LI>
<LI>CAT III: provide information that potentially could lead to compromise.</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

Name: STIG 3.5
ID: DD18E81F-3507-41FA-9DFA-2A9A15B5479F
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APP<I>ID</I>: CAT <I>SEV</I>]. DISA STIG identifies several severities with respect to vulnerabilities:

<LI>CAT I: allow an attacker immediate access into a machine, allow super user access, or bypass a firewall.</LI>
<LI>CAT II: provide information that have a high potential of giving access to an intruder.</LI>
<LI>CAT III: provide information that potentially could lead to compromise.</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

Name: STIG 3.6
ID: 000CA760-0FED-4374-8AA2-6FA3968A07B1
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APP<I>ID</I>: CAT <I>SEV</I>]. DISA STIG identifies several severities with respect to vulnerabilities:

<LI>CAT I: allow an attacker immediate access into a machine, allow super user access, or bypass a firewall.</LI>
<LI>CAT II: provide information that have a high potential of giving access to an intruder.</LI>
<LI>CAT III: provide information that potentially could lead to compromise.</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APP5080: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APP5100: CAT II is not considered "In Place".

Name: STIG 3.7
ID: E69C07C0-81D8-4B04-9233-F3E74167C3D2
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APP<I>ID</I>: CAT <I>SEV</I>]. DISA STIG identifies several severities with respect to vulnerabilities:

<LI>CAT I: allow an attacker immediate access into a machine, allow super user access, or bypass a firewall.</LI>
<LI>CAT II: provide information that have a high potential of giving access to an intruder.</LI>
<LI>CAT III: provide information that potentially could lead to compromise.</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APP5080: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APP5100: CAT II is not considered "In Place".

Name: STIG 3.9
ID: 1A9D736B-2D4A-49D1-88CA-DF464B40D732
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APP<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APP5080: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APP5100: CAT II is not considered "In Place".

Name: STIG 4.1
ID: 95227C50-A9E4-4C9D-A8AF-FD98ABAE1F3C
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.10
ID: EF1FF442-1673-4CF1-B7C4-920F1A96A8150
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.2
ID: 672C15F8-8822-4E05-8C9E-1A4BAAA7A373
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.3
ID: A0B313F0-29BD-430B-9E34-6D10F1178506
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>

<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.4
ID: ECEC5CA2-7ACA-4B70-BF44-3248B9C6F4F8
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.5
ID: E6010E0A-7F71-4388-B8B7-EE9A02143474
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.6
ID: EFB9B012-44D6-456D-B197-03D2FD7C7AD6
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>].

DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.7
ID: B04A1E01-F1C1-48D3-A827-0F70872182D7
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>].
DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.8
ID: E6805D9F-D5B5-4192-962C-46828FF68507
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>].
DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: STIG 4.9

ID: 7B9F7B3B-07FC-4B61-99A1-70E3BB23A6A0
Each requirement or recommendation identified by the Defense Information Systems Agency (DISA) STIG is represented by a STIG identifier (STIGID), which corresponds to a checklist item and a severity code [APSC-DV-<I>ID</I>: CAT <I>SEV</I>]. DISA STIG defines three severities with respect to vulnerabilities where their:

<LI>exploitation leads to direct and immediate loss of Confidentiality, Availability, or Integrity (CAT I).</LI>
<LI>exploitation potentially results in loss of Confidentiality, Availability, or Integrity (CAT II).</LI>
<LI>existence degrades protections against loss of Confidentiality, Availability, or Integrity (CAT III).</LI> </UL>

The following table summarizes the number of issues identified across the different STIGIDs broken down by Fortify Priority Order. The status of a STIGID is considered "In Place" when there are no issues reported for a given STIGID.

If the project is missing a Fortify Static Code Analyzer (SCA) scan, or the scan contains findings that have not been fixed, hidden or suppressed, STIGID APSC-DV-003170: CAT II is not considered "In Place". Similarly, if the project is missing a Fortify WebInspect scan, or the scan contains any critical findings, STIGID APSC-DV-001460: CAT II and STIGID APSC-DV-002930: CAT II are not considered "In Place".

Name: WASC 2.00
ID: 74f8081d-dd49-49da-880f-6830cebe9777
The Web Application Security Consortium (WASC) was created as a cooperative effort to standardize, clarify, and organize the threats to the security of a web site. Version 2.00 of their Threat Classification outlines the attacks and weaknesses that can commonly lead to a website being compromised.

Name: WASC 24 + 2
ID: 9DC61E7F-1A48-4711-BBFD-E9DFF537871F
The Web Application Security Consortium (WASC) was created as a cooperative effort to standardize, clarify, and organize the threats to the security of a web site.

Properties

WinForms.CollectionMutationMonitor.Label=WinFormsDataSource
awt.toolkit=sun.awt.X11.XToolkit
com.fortify.AuthenticationKey=/home/pgupta25/.fortify/config/tools
com.fortify.Core=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core
com.fortify.InstallRoot=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0
com.fortify.InstallationUserName=pgupta25
com.fortify.SCAExecutablePath=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/bin/sourceanalyzer
com.fortify.TotalPhysicalMemory=8363917312
com.fortify.VS.RequireASPPrecompilation=true
com.fortify.WorkingDirectory=/home/pgupta25/.fortify
com.fortify.locale=en
com.fortify.sca.AddImpliedMethods=true
com.fortify.sca.AntCompilerClass=com.fortify.dev.ant.SCACompiler
com.fortify.sca.AppendLogFile=true
com.fortify.sca.BuildID=owa
com.fortify.sca.BundleControlflowIssues=true
com.fortify.sca.BytecodePreview=true
com.fortify.sca.CollectPerformanceData=true
com.fortify.sca.CustomRulesDir=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/config/customrules
com.fortify.sca.DaemonCompilers=com.fortify.sca.util.compilers.GppCompiler,com.fortify.sca.util.compilers.GccCompiler,com.f

ortify.sca.util.compilers.AppleGppCompiler,com.fortify.sca.util.compilers.AppleGccCompiler,com.fortify.sca.util.compilers.Micr
osoftCompiler,com.fortify.sca.util.compilers.MicrosoftLinker,com.fortify.sca.util.compilers.LdCompiler,com.fortify.sca.util.com
pilers.ArUtil,com.fortify.sca.util.compilers.SunCCompiler,com.fortify.sca.util.compilers.SunCppCompiler,com.fortify.sca.util.co
mpilers.IntelCompiler,com.fortify.sca.util.compilers.ExternalCppAdapter,com.fortify.sca.util.compilers.ClangCompiler
com.fortify.sca.DeadCodeFilter=true
com.fortify.sca.DeadCodeIgnoreTrivialPredicates=true
com.fortify.sca.DefaultAnalyzers=semantic:dataflow:controlflow:nullptr:configuration:content:structural:buffer
com.fortify.sca.DefaultFileTypes=java,rb,erb,jsp,jspx,jspf,tag,tagx,tld,sql,cfm,php,phtml,ctp,pks,pkh,pkb,xml,config,Config,settin
gs,properties,dll,exe,winmd,cs,vb,asax,ascx,ashx,asmx,aspx,master,Master,xaml,baml,cshtml,vbhtml,inc,asp,vbscript,js,ini,bas,cls
,vbs,frm,ctl,html,htm,xsd,wsdd,xmi,py,cfml,cfc,abap,xhtml,cpx,xcfg,jsff,as,mxml,cbl,cscfg,csdef,wadcfg,wadcfgx,appxmanifest,
wsdl,plist,bsp,ABAP,BSP,swift,page,trigger,scala,ts,conf,json,yaml,yml
com.fortify.sca.DefaultJarsDirs=default_jars
com.fortify.sca.DefaultRulesDir=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/config/rules
com.fortify.sca.DisableDeadCodeElimination=false
com.fortify.sca.DisableFunctionPointers=false
com.fortify.sca.DisableGlobals=false
com.fortify.sca.DisableInferredConstants=false
com.fortify.sca.EnableInterproceduralConstantResolution=true
com.fortify.sca.EnableNestedWrappers=true
com.fortify.sca.EnableStructuralMatchCache=true
com.fortify.sca.EnableWrapperDetection=true
com.fortify.sca.FVDLDisableDescriptions=false
com.fortify.sca.FVDLDisableProgramData=false
com.fortify.sca.FVDLDisableSnippets=false
com.fortify.sca.FVDLStylesheet=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/resources/sca/fvdl2html.xsl
com.fortify.sca.IndirectCallGraphBuilders=WinFormsAdHocFunctionBuilder,VirtualCGBuilder,J2EEIndirectCGBuilder,JNICG
Builder,StoredProcedureResolver,JavaWSCGBuilder,StrutsCGBuilder,DotNetWSCGBuilder,SqlServerSPResolver,ASPCGBuild
er,ScriptedCGBuilder,NewJspCustomTagCGBuilder,DotNetCABCGBuilder,StateInjectionCGBuilder,SqlServerSPResolver2,PH
PLambdaResolver,JavaWebCGBuilder
com.fortify.sca.JVMArgs=-XX:SoftRefLRUPolicyMSPerMB=3000 -Xmx4096M -Xss16M
com.fortify.sca.JavaSourcepathSearch=true
com.fortify.sca.JdkVersion=1.8
com.fortify.sca.LogFileDir=/home/pgupta25/.fortify/sca19.1/log
com.fortify.sca.LogFileExt=.log
com.fortify.sca.LogFileName=sca.log
com.fortify.sca.LogFileNameNoExt=sca
com.fortify.sca.LogFilePath=/home/pgupta25/.fortify/sca19.1/log/sca.log
com.fortify.sca.LogLevel=INFO
com.fortify.sca.LowSeverityCutoff=1.0
com.fortify.sca.MachineOutputMode=
com.fortify.sca.MultithreadedAnalysis=true
com.fortify.sca.NoNestedOutTagOutput=org.apache.taglibs.standard.tag.rt.core.RemoveTag,org.apache.taglibs.standard.tag.rt.cor
e.SetTag
com.fortify.sca.OldVbNetExcludeFileTypes=vb,asax,ascx,ashx,asmx,aspx,xaml,cshtml,vbhtml
com.fortify.sca.PID=11572
com.fortify.sca.Phase0HigherOrder.Languages=python,ruby,swift,javascript,typescript
com.fortify.sca.Phase0HigherOrder.Level=1
com.fortify.sca.PrintPerformanceDataAfterScan=false
com.fortify.sca.ProjectRoot=/home/pgupta25/.fortify
com.fortify.sca.ProjectRoot=/home/pgupta25/.fortify
com.fortify.sca.Renderer=fpr

com.fortify.sca.RequireMapKeys=classrule
com.fortify.sca.ResultsFile=/srv/openmrs_code/org/openmrs/module/owa/owa_scan.fpr
com.fortify.sca.SolverTimeout=15
com.fortify.sca.SqlLanguage=PLSQL
com.fortify.sca.SuppressLowSeverity=true
com.fortify.sca.ThreadCount.NameTableLoading=1
com.fortify.sca.TypeInferenceFunctionTimeout=60
com.fortify.sca.TypeInferenceLanguages=javascript,typescript,python,ruby
com.fortify.sca.TypeInferencePhase0Timeout=300
com.fortify.sca.UnicodeInputFile=true
com.fortify.sca.UniversalBlacklist=.*yyparse.*
com.fortify.sca.alias.mode.csharp=fs
com.fortify.sca.alias.mode.javascript=fi
com.fortify.sca.alias.mode.scala=fi
com.fortify.sca.alias.mode.swift=fi
com.fortify.sca.alias.mode.typescript=fi
com.fortify.sca.alias.mode.vb=fs
com.fortify.sca.analyzer.controlflow.EnableLivenessOptimization=false
com.fortify.sca.analyzer.controlflow.EnableMachineFiltering=false
com.fortify.sca.analyzer.controlflow.EnableRefRuleOptimization=false
com.fortify.sca.analyzer.controlflow.EnableTimeOut=true
com.fortify.sca.compilers.ant=com.fortify.sca.util.compilers.AntAdapter
com.fortify.sca.compilers.ar=com.fortify.sca.util.compilers.ArUtil
com.fortify.sca.compilers.armcc=com.fortify.sca.util.compilers.ArmCcCompiler
com.fortify.sca.compilers.armcpp=com.fortify.sca.util.compilers.ArmCppCompiler
com.fortify.sca.compilers.c++=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.compilers.cc=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.compilers.clearmake=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.compilers.fortify=com.fortify.sca.util.compilers.FortifyCompiler
com.fortify.sca.compilers.g++=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.compilers.g++-*=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.compilers.g++2*=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.compilers.g++3*=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.compilers.g++4*=com.fortify.sca.util.compilers.GppCompiler
com.fortify.sca.compilers.gcc=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.compilers.gcc-*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.compilers.gcc2*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.compilers.gcc3*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.compilers.gcc4*=com.fortify.sca.util.compilers.GccCompiler
com.fortify.sca.compilers.gmake=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.compilers.gradle=com.fortify.sca.util.compilers.GradleAdapter
com.fortify.sca.compilers.gradlew=com.fortify.sca.util.compilers.GradleAdapter
com.fortify.sca.compilers.icc=com.fortify.sca.util.compilers.IntelCompiler
com.fortify.sca.compilers.icpc=com.fortify.sca.util.compilers.IntelCompiler
com.fortify.sca.compilers.jam=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.compilers.javac=com.fortify.sca.util.compilers.JavacCompiler
com.fortify.sca.compilers.ld=com.fortify.sca.util.compilers.LdCompiler
com.fortify.sca.compilers.make=com.fortify.sca.util.compilers.TouchlessCompiler
com.fortify.sca.compilers.mvn=com.fortify.sca.util.compilers.MavenAdapter
com.fortify.sca.compilers.scalac=com.fortify.sca.util.compilers.ScalacCompiler
com.fortify.sca.compilers.tcc=com.fortify.sca.util.compilers.ArmCcCompiler

com.fortify.sca.compilers.tcpp=com.fortify.sca.util.compilers.ArmCppCompiler
com.fortify.sca.compilers.touchless=com.fortify.sca.util.compilers.FortifyCompiler
com.fortify.sca.cpfe.441.command=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/private-bin/sca/cpfe441.rfct
com.fortify.sca.cpfe.command=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/private-bin/sca/cpfe48
com.fortify.sca.cpfe.file.option=--gen_c_file_name
com.fortify.sca.cpfe.options=--remove_unneeded_entities --suppress_vtbl -tused
com.fortify.sca.cpfe.options=--remove_unneeded_entities --suppress_vtbl -tused
com.fortify.sca.env.exesearchpath=/sbin:/bin:/usr/bin:/usr/local/bin
com.fortify.sca.fileextensions.ABAP=ABAP
com.fortify.sca.fileextensions.BSP=ABAP
com.fortify.sca.fileextensions.Config=XML
com.fortify.sca.fileextensions.abap=ABAP
com.fortify.sca.fileextensions.appxmanifest=XML
com.fortify.sca.fileextensions.as=ACTIONSCRIPT
com.fortify.sca.fileextensions.asp=ASP
com.fortify.sca.fileextensions.bas=VB6
com.fortify.sca.fileextensions.bsp=ABAP
com.fortify.sca.fileextensions.cfc=CFML
com.fortify.sca.fileextensions.cfm=CFML
com.fortify.sca.fileextensions.cfml=CFML
com.fortify.sca.fileextensions.cls=VB6
com.fortify.sca.fileextensions.conf=HOCON
com.fortify.sca.fileextensions.config=XML
com.fortify.sca.fileextensions.cpx=XML
com.fortify.sca.fileextensions.cscfg=XML
com.fortify.sca.fileextensions.csdef=XML
com.fortify.sca.fileextensions.ctl=VB6
com.fortify.sca.fileextensions.ctp=PHP
com.fortify.sca.fileextensions.erb=RUBY_ERB
com.fortify.sca.fileextensions.faces=JSPX
com.fortify.sca.fileextensions.frm=VB6
com.fortify.sca.fileextensions.htm=HTML
com.fortify.sca.fileextensions.html=HTML
com.fortify.sca.fileextensions.ini=JAVA_PROPERTIES
com.fortify.sca.fileextensions.java=JAVA
com.fortify.sca.fileextensions.js=TYPESCRIPT
com.fortify.sca.fileextensions.jsff=JSPX
com.fortify.sca.fileextensions.json=JSON
com.fortify.sca.fileextensions.jsp=JSP
com.fortify.sca.fileextensions.jspf=JSP
com.fortify.sca.fileextensions.jspx=JSPX
com.fortify.sca.fileextensions.jsx=TYPESCRIPT
com.fortify.sca.fileextensions.mxml=MXML
com.fortify.sca.fileextensions.page=VISUAL_FORCE
com.fortify.sca.fileextensions.php=PHP
com.fortify.sca.fileextensions.phtml=PHP
com.fortify.sca.fileextensions.pkb=PLSQL
com.fortify.sca.fileextensions.pkh=PLSQL
com.fortify.sca.fileextensions.pks=PLSQL
com.fortify.sca.fileextensions.plist=XML
com.fortify.sca.fileextensions.properties=JAVA_PROPERTIES

com.fortify.sca.fileextensions.py=PYTHON
com.fortify.sca.fileextensions.rb=RUBY
com.fortify.sca.fileextensions.scala=SCALA
com.fortify.sca.fileextensions.settings=XML
com.fortify.sca.fileextensions.sql=SQL
com.fortify.sca.fileextensions.swift=SWIFT
com.fortify.sca.fileextensions.tag=JSP
com.fortify.sca.fileextensions.tagx=JSP
com.fortify.sca.fileextensions.tld=TLD
com.fortify.sca.fileextensions.trigger=APEX_TRIGGER
com.fortify.sca.fileextensions.ts=TYPESCRIPT
com.fortify.sca.fileextensions.tsx=TYPESCRIPT
com.fortify.sca.fileextensions.vbs=VBSCRIPT
com.fortify.sca.fileextensions.vbscript=VBSCRIPT
com.fortify.sca.fileextensions.wadcfg=XML
com.fortify.sca.fileextensions.wadcfgx=XML
com.fortify.sca.fileextensions.wsdd=XML
com.fortify.sca.fileextensions.wsdl=XML
com.fortify.sca.fileextensions.xcfg=XML
com.fortify.sca.fileextensions.xhtml=JSPX
com.fortify.sca.fileextensions.xmi=XML
com.fortify.sca.fileextensions.xml=XML
com.fortify.sca.fileextensions.xsd=XML
com.fortify.sca.fileextensions.yaml=YAML
com.fortify.sca.fileextensions.yml=YAML
com.fortify.sca.jsp.UseNativeParser=true
com.fortify.sca.parser.python.ignore.module.1=test.badsyntax_future3
com.fortify.sca.parser.python.ignore.module.2=test.badsyntax_future4
com.fortify.sca.parser.python.ignore.module.3=test.badsyntax_future5
com.fortify.sca.parser.python.ignore.module.4=test.badsyntax_future6
com.fortify.sca.parser.python.ignore.module.5=test.badsyntax_future7
com.fortify.sca.parser.python.ignore.module.6=test.badsyntax_future8
com.fortify.sca.parser.python.ignore.module.7=test.badsyntax_future9
com.fortify.sca.parser.python.ignore.module.8=test.badsyntax_nocaret
com.fortify.sca.skip.libraries.AngularJS=angular.js,angular.min.js,angular-animate.js,angular-aria.js,angular_1_router.js,angular-cookies.js,angular-message-format.js,angular-messages.js,angular-mocks.js,angular-parse-ext.js,angular-resource.js,angular-route.js,angular-sanitize.js,angular-touch.js
com.fortify.sca.skip.libraries.ES6=es6-shim.min.js,system-polyfills.js,shims_for_IE.js
com.fortify.sca.skip.libraries.jQuery=jquery.js,jquery.min.js,jquery-migrate.js,jquery-migrate.min.js,jquery-ui.js,jquery-ui.min.js,jquery.mobile.js,jquery.mobile.min.js,jquery.color.js,jquery.color.min.js,jquery.color.svg-names.js,jquery.color.svg-names.min.js,jquery.color.plus-names.js,jquery.color.plus-names.min.js,jquery.tools.min.js
com.fortify.sca.skip.libraries.javascript=bootstrap.js,bootstrap.min.js,typescript.js,typescriptServices.js
com.fortify.sca.skip.libraries.typescript=typescript.d.ts,typescriptServices.d.ts
com.fortify.search.defaultSyntaxVer=2
com.sun.management.jmxremote=true
file.encoding=UTF-8
file.encoding.pkg=sun.io
file.separator=/
java.awt.graphicsenv=sun.awt.X11GraphicsEnvironment
java.awt.headless=true
java.awt.printerjob=sun.print.PSPrinterJob

java.class.path=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/lib/exe/sca-exe.jar
java.class.version=52.0
java.endorsed.dirs=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/endorsed
java.ext.dirs=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/ext:/usr/java/packages/lib/ext
java.home=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre
java.io.tmpdir=/tmp
java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
java.rmi.server.randomIDs=true
java.runtime.name=OpenJDK Runtime Environment
java.runtime.version=1.8.0_181-b02
java.specification.name=Java Platform API Specification
java.specification.vendor=Oracle Corporation
java.specification.version=1.8
java.vendor=Azul Systems, Inc.
java.vendor.url=http://www.azulsystems.com/
java.vendor.url.bug=http://www.azulsystems.com/support/
java.version=1.8.0_181
java.vm.info=mixed mode
java.vm.name=OpenJDK 64-Bit Server VM
java.vm.specification.name=Java Virtual Machine Specification
java.vm.specification.vendor=Oracle Corporation
java.vm.specification.version=1.8
java.vm.vendor=Azul Systems, Inc.
java.vm.version=25.181-b02
line.separator=

log4j.configurationFile=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/config/log4j2.xml
log4j.isThreadContextMapInheritable=true
max.file.path.length=255
os.arch=amd64
os.name=Linux
os.version=4.15.0-58-generic
path.separator=:
stderr.isatty=false
stdout.isatty=false
sun.arch.data.model=64
sun.boot.class.path=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/resources.jar:/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/rt.jar:/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/sunrsasign.jar:/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/jsse.jar:/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/jce.jar:/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/charsets.jar:/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/jfr.jar:/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/classes
sun.boot.library.path=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/jre/lib/amd64
sun.cpu.endian=little
sun.cpu.isalist=
sun.io.unicode.encoding=UnicodeLittle
sun.java.command=sourceanalyzer -Djava.awt.headless=true -Dcom.sun.management.jmxremote=true -XX:SoftRefLRUPolicyMSPerMB=3000 -Dcom.fortify.sca.env.exesearchpath=/sbin:/bin:/usr/bin:/usr/local/bin -Dcom.fortify.sca.ProjectRoot=/home/pgupta25/.fortify -Dstdout.isatty=false -Dstderr.isatty=false -Dcom.fortify.sca.PID=11572 -Xmx4096M -Dcom.fortify.TotalPhysicalMemory=8363917312 -Xss16M -Dcom.fortify.sca.JVMArgs=-XX:SoftRefLRUPolicyMSPerMB=3000 -Xmx4096M -Xss16M -Djava.class.path=/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/lib/exe/sca-exe.jar -scan @/home/pgupta25/.fortify/Eclipse.Plugin-19.1.0/owa/owaScan.txt

sun.jnu.encoding=UTF-8
sun.management.compiler=HotSpot 64-Bit Tiered Compilers
sun.os.patch.level=unknown
user.country=US
user.dir=/home/pgupta25
user.home=/home/pgupta25
user.language=en
user.name=pgupta25
user.timezone=America/New_York

Commandline Arguments

-scan
-b
owa
-format
fpr
-machine-output
-f
/srv/openmrs_code/org/openmrs/module/owa/owa_scan.fpr

Warnings

[1001] Unexpected exception while parsing file (javascript) /srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/resources/javascript/main.js: Parse error at line 69, column 1. Encountered: var
[1001] Unexpected exception while parsing file (javascript) /srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/resources/javascript/main.js: Parse error at line 69, column 1. Encountered: var
[1001] Unexpected exception while parsing file (javascript) /srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/resources/javascript/main.js: Parse error at line 69, column 1. Encountered: var
[12002] Could not locate the deployment descriptor (web.xml) for your web application. Please build your web application and try again. File: /srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/manage.jsp
[12003] Assuming Java source level to be 1.8 as it was not specified. Note that the default value may change in future versions.
[12004] The Java frontend was unable to resolve the following include:
/WEB-INF/template/include.jsp at /srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/settings.jsp:1.
/WEB-INF/template/footer.jsp at /srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/settings.jsp:58.
/WEB-INF/template/header.jsp at /srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/settings.jsp:3.
[12019] The following references to Java functions could not be resolved. These functions may be part of classes that could not be found, or there may be a type error at the call site of the given function relative to the function declaration. Please ensure the Java source code can be compiled by a Java compiler.
javax.servlet.http.HttpSession.getAttribute
javax.servlet.http.HttpSession.removeAttribute

[12022] The class "javax.servlet.http.HttpServlet" could not be found on the classpath, but it was found in the JAR file provided by Fortify in "/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/default_jars/javax.servlet-api-3.0.1.jar" as a convenience. To ensure consistent translation behavior add the JAR file that contains "javax.servlet.http.HttpServlet" to the classpath given to the translation step. Refer to the documentation about "default JARs" in the SCA User Guide for more information.

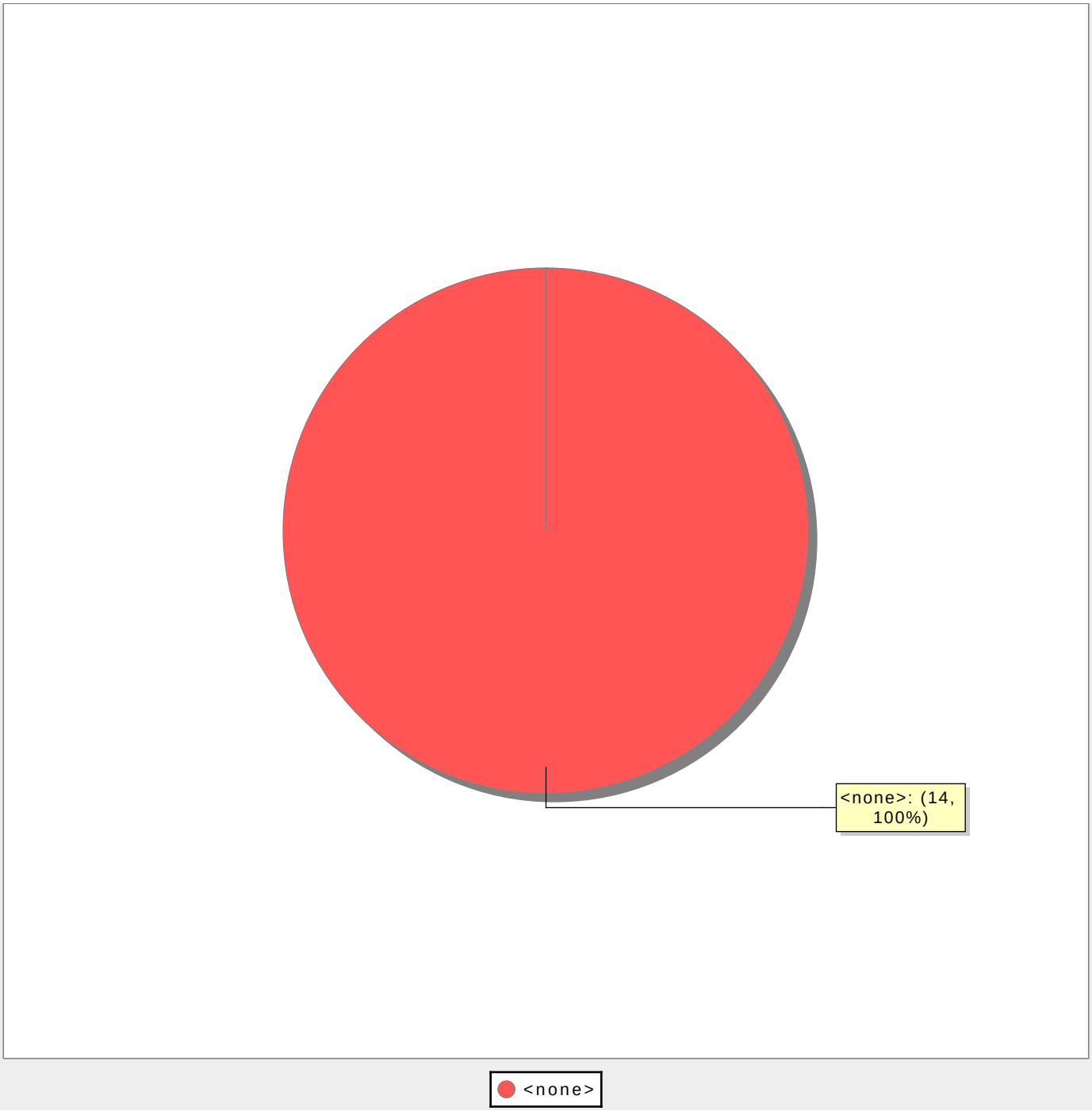
[12022] The class "javax.servlet.jsp.PageContext" could not be found on the classpath, but it was found in the JAR file provided by Fortify in "/opt/Fortify/Fortify_SCA_and_Apps_19.1.0/Core/default_jars/javax.servlet.jsp-api.jar" as a convenience. To ensure consistent translation behavior add the JAR file that contains "javax.servlet.jsp.PageContext" to the classpath given to the translation step. Refer to the documentation about "default JARs" in the SCA User Guide for more information.
[1214] Multiple definitions found for class /manage.jsp
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/manage.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/manage.jsp).
[1214] Multiple definitions found for class JSPPAGE._/_jspmanage_jsp$ftfy_frameworkVisibleObjects
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/manage.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/manage.jsp).
[1214] Multiple definitions found for class /settings.jsp
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/settings.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/settings.jsp).
[1214] Multiple definitions found for class JSPPAGE._/_jspsettings_jsp$ftfy_frameworkVisibleObjects
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/settings.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/settings.jsp).
[1214] Multiple definitions found for class /localHeader.jsp
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/template/localHeader.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/template/localHeader.jsp).
[1214] Multiple definitions found for class JSPPAGE._/_jsplocalHeader_jsp$ftfy_frameworkVisibleObjects
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/template/localHeader.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/classes/web/module/template/localHeader.jsp).
[1214] Multiple definitions found for class /manage.jsp
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/manage.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/manage.jsp).
[1214] Multiple definitions found for class JSPPAGE._/_jspmanage_jsp$ftfy_frameworkVisibleObjects
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/manage.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/manage.jsp).
[1214] Multiple definitions found for class /settings.jsp
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/settings.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/settings.jsp).
[1214] Multiple definitions found for class JSPPAGE._/_jspsettings_jsp$ftfy_frameworkVisibleObjects
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/settings.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/settings.jsp).
[1214] Multiple definitions found for class /localHeader.jsp
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/template/localHeader.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/template/localHeader.jsp).
[1214] Multiple definitions found for class JSPPAGE._/_jsplocalHeader_jsp$ftfy_frameworkVisibleObjects
(/srv/openmrs_code/org/openmrs/module/owa/omod/src/main/webapp/template/localHeader.jsp and
/srv/openmrs_code/org/openmrs/module/owa/omod/target/owa-1.10.0/web/module/template/localHeader.jsp).
[1215] Could not locate the root (WEB-INF) of the web application. Please build your web application and try again.
[1216] Unable to locate a class for import org.openmrs.web.WebConstants
[1237] The following references to Java symbols could not be resolved. Please make sure to supply all the required JAR files that contain these symbols to SCA.
java.lang.String.OPENMRS_ERROR_ATTR
java.lang.String.OPENMRS_MSG_ATTR

Issue Count by Category

Issues by Category	
Cross-Site Scripting: DOM	6
Header Manipulation	3
File Disclosure: J2EE	2
Open Redirect	1
Path Manipulation	1
Race Condition: Singleton Member Field	1

Issue Breakdown by Analysis

Issues by Analysis



New Issues

Issues by New Issue

The following issues have been discovered since the last scan.