AVGRADERT
Dato:.24.05.2019
Sign: Bjørn Berg (digitalt)

BEGRENSET
iht sikkerhetsloven § 11 og § 12
jf offentleglova § 13

Rapport

Arkitektur- og kildekodegjennomgang EVA Resultat

Valgdirektoratet

Versjon 1.0
13.6.2018

1 Sammendrag

mnemonic har gjennomført en kodegjennomgang av de ulike delene av EVA applikasjonen. Denne rapporten beskriver funn og forbedringsforslag for EVA Resultat. Rapporten er hovedsaklig basert på funn fra analyseverktøyet MicroFocus Fortify SCA.

2 Om deloppdraget

For denne komponenten fikk vi i oppstartsmøtet følgende ønske om spissing av oppdraget:

Eva Resultat 10%:

- *Sentralt driftet web-applikasjon – ingen brukere*
- *Sikring gjennom infrastruktur*
- *Kodekvalitet*

Analyse fra kildekodeverktøy har tatt mesteparten av tiden for dette oppdraget.

mnemonic
- securing your business

3 Generelt om applikasjonen

Dette kapittelet inneholder observasjoner knyttet til kvalitet og arkitektur. Dette er basert både på vår jobbing med koden og få den til å kompilere og bygge, samt dypere analyser av strukturen og potensielle problemer vi ser med tanke på vedlikehold, videreutvikling og rene sikkerhetsaspekter.

3.1 Bygging av kode

Det oppsto noen problemer i forbindelse med bygging av koden med Maven. For å kompilere koden var vi nødt til å ekskludere all testing av kode, både enhetstester og integrasjonstester. Dette kan være et tegn på altfor sterk kobling mellom koden i seg selv og testrammeverket. Generelt kan sies at enhetstester bør lages slik at de ulike enhetene testes hver for seg uberoende av andre enheter.
Anbefaling 1 Se over eventuelle avhengigheter mellom applikasjons- og testkode

Versjonen som er blitt brukt for de ulike komponentene i Spring Framework er ikke konsekvent. Spesielt er det avhengighet til en eldre versjon av `org.springframework.integration:springintegration-core` enn de andre komponentene. Versjonsangivelsen til denne var i tillegg satt eksplitt i den aktuelle Maven-avhengigheten, mens de andre bruker en Maven-variabel («property»), noe som kan være grunnen til at denne har blitt hengende igjen.

Anbefaling 2 Bruk samme versjon for alle komponenter innenfor samme biblioteksgruppe

4 Verktøybaserte funn: komponenter med sårbarheter

Dette kapittelet inneholder funn fra OWASP Dependency Check-verktøyet. Se vedlagte rårapporter for å få mer informasjon om hvilke versjoner som ikke er sårbare, hvilke CVE (Common Vulnerability Enumeration)-sårbarheter det dreier seg om, m.m.

Følgende Java-bibliotek er i bruk i koden og har kjente sårbarheter:

- org.springframework:spring-core:5.0.1.RELEASE

5 Verktøybaserte funn: kildekodefunn

Dette kapittelet inneholder detaljert informasjon om hvert enkelt funn, slik Fortify SCA rapporterer det.

Vi har prøvd å kvalitetssikre funnene ved å fjerne de som er falske positive. Det er dog ikke alltid lett å komme utenfra å forstå koden og det vil nok være funn som vi har tatt med, men som utviklerne vil kategorisere som falske positive. De færreste funnene er direkte utnyttbare, men vi mener at ved å følge anbefalingene vil det være mindre fare for sikkerhetsrelaterte hendelser, og vi bruker her «sikkerhet i dybden» som prinsipp. Funnene vi anser som reelle har vi kategorisert iht Fortifys 4 innebygde analysekategorier:

- Reliability Issue – Dette er svakheter som kan gjøre at applikasjonen ikke oppfører seg som den skal. Det er ofte relatert til fysiske begrensninger og gjelder gjern filaksess, nettverkskommunikasjon, minneproblemer, o.l.
- Bad Practice – Denne kategorien inneholder både sikkerhetsrelaterte sårbarheter og kvalitetsrelaterte.
- Suspicious – Her har vi funn som vi enten er usikre på om er utnyttbare, eller som vil kunne være én av flere faktorer som skal til for at en sårbarhet oppstår, mtp sikkerhet i dybden.

- Exploitable – Dette er funn som er direkte utnyttbare. Det kan være manglende validering og outputkoding (Injection-/Cross-Site Scripting-angrep), manglende transportsikring (klartekstkommunikasjon i nettverk), eller andre angrep som vil kunne skje uten at andre sikkerhetsmekanismer brytes. Det vil dog være ulikt hvor alvorlige disse er, f.eks. om det krever aksess til lokalt nettverk eller om det kan gjøres direkte fra Internett.

Et funn kan høre hjemme i flere kategorier, og det er særlig «Suspicious» hvor vi er usikre på om sårbarheten er reell eller ikke.

Funnene er sortert etter alvorlighetsgrad slik at de mest kritiske funnene kommer først.

[High] Cross-Site Scripting / Persistent

Input Validation and Representation

Analysis: Suspicious

Abstract

Sending unvalidated data to a web browser 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 Persistent (also known as Stored) XSS, the untrusted source is typically a database or other back-end data store, while in the case of Reflected XSS it is typically a web request.
2. The data is included in dynamic content that is sent to a web user without being validated.

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 may execute. 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 JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

```
<%...
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp where id="+eid); if (rs != null) {    rs.next();
   String name = rs.getString("name");
}
%>


Employee Name: <%= name %>
```

This code functions correctly when the values of `name` are well-behaved, but it does nothing to prevent exploits if they are not. This code can appear less dangerous because the value of `name` is read from a database, whose contents are apparently managed by the application. However, if the value of `name` originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker may execute malicious commands in the user's web browser. This type of exploit, known as Persistent (or Stored) XSS, is particularly insidious because the indirection caused by the data store makes it more difficult to identify the threat and increases the possibility that the attack will affect multiple users. XSS got its start in this form with web sites that offered a "guestbook" to visitors. Attackers would include

JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code.

Example 2: The following JSP code segment reads an employee ID, `eid`, from an HTTP request and displays it to the user.

```
<% String eid = request.getParameter("eid"); %> ...
Employee ID: <%= eid %>
```

As in Example 1, this code operates correctly if `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.

Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use e-mail 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.

Some think that in the mobile world, classic web application vulnerabilities, such as cross-site scripting, 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 enables JavaScript in Android's WebView (by default, JavaScript is disabled) and loads a page based on the value received from an Android intent.

```
...
        WebView webview = (WebView) findViewById(R.id.webview);
webview.getSettings().setJavaScriptEnabled(true);
        String url = this.getIntent().getExtras().getString("url");         webview.loadUrl(url); ...
```

If the value of `url` starts with `javascript:`, JavaScript code that follows will execute within the context of the web page inside WebView.

As the examples demonstrate, 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:

- As in Example 1, 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.

- As in Example 2, 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 e-mailed 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.

- As in Example 3, 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.

Recommendation

The solution to XSS is to ensure that validation occurs in the correct places and checks 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 above 128 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).

Once 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.

5.1 visRapport.jsp

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 41
Action OutCall getInputStream(return)
Rule id E080F8D1-F512-49DA-B076-CBA2A217C538
Fact Call Direct : java.net.URLConnection.getInputStream

```
URL rapport = new URL(rapporturl);
HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
```

```
 40|conn.connect();
>41|BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));  42|String line;
 43|while ((line = br.readLine()) != null) {
 44|    xmlFormatert.append(line).append('\n');
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 120
Action InCall print(0)
Rule id 4D69E4E2-959C-4EF8-84F4-A6489B288ABF
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags NO_NEW_LINE, WEBSERVICE, XSS

```
 117|        <br>
 118|        <br>
 119|        <pre>
>120|        <%=xmlFormatert.toString() %>
 121|        </pre>
 122|    </div>
 123|</div>
```

[High] Cross-Site Scripting / Reflected

Input Validation and Representation

Analysis: Suspicious

Abstract

Sending unvalidated data to a web browser 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 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.

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 may execute. 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 JSP code segment reads an employee ID, `eid`, from an HTTP request and displays it to the user.

```
<% String eid = request.getParameter("eid"); %> ...
Employee ID: <%= eid %>
```

The code in this example operates correctly if `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.

Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use e-mail 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.

Example 2: The following JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

```
<%...
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp where id="+eid); if (rs != null) {    rs.next();
   String name = rs.getString("name");
}
%>


Employee Name: <%= name %>
```

As in Example 1, this code functions correctly when the values of `name` are well-behaved, but it does nothing to prevent exploits if they are not. Again, this code can appear less dangerous because the value of `name` is read from a database, whose contents are apparently managed by the application. However, if the value of `name` originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker may execute malicious commands in the user's web browser. This type of exploit, known as Persistent (or Stored) XSS, is particularly insidious because the indirection caused by the data store makes it more difficult to identify the threat and increases the possibility that the attack will affect multiple users. XSS got its start in this form with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code.

Some think that in the mobile world, classic web application vulnerabilities, such as cross-site scripting, 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 enables JavaScript in Android's WebView (by default, JavaScript is disabled) and loads a page based on the value received from an Android intent.

```
...
        WebView webview = (WebView) findViewById(R.id.webview);
webview.getSettings().setJavaScriptEnabled(true);
        String url = this.getIntent().getExtras().getString("url");        webview.loadUrl(url); ...
```

If the value of `url` starts with `javascript:`, JavaScript code that follows will execute within the context of the web page inside WebView.

As the examples demonstrate, 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:

- As in Example 1, 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 e-mailed 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.

- As in Example 2, 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.

- As in Example 3, 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.

Recommendation

The solution to XSS is to ensure that validation occurs in the correct places and checks 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 above 128 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).

Once 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.

5.2 endreSkjemastatus.jsp

Issue 255F1A7D573185F0361E3DAC9F61FBA2:

Source :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 22
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
 19|</head>
 20|<body>
 21|<% if (ok) { %>
>22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
 24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status  25|har endret seg.
```

Sink :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 22
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags PRIMARY_KEY, WEB, XSS

```
 19|</head>
 20|<body>
 21|<% if (ok) { %>
>22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
 24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status  25|har endret seg.
```

Issue 255F1A7D573185F0361E3DAC9F61FBA3:

Source :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 24
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
 21|<% if (ok) { %>
 22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
>24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status
 25|har endret seg.
 26|<% } %>
 27|</body>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 24
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792 **Fact** Call Direct :
javax.servlet.jsp.JspWriter.print

Fact TaintFlags PRIMARY_KEY, WEB, XSS

```
 21|<% if (ok) { %>
 22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
>24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status
 25|har endret seg.
 26|<% } %>
 27|</body>
```

Issue FFD1B39F92A685E3710F537D9B2CE9AB:

Source :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 22
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
 19|</head>
 20|<body>
 21|<% if (ok) { %>
>22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
 24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status  25|har endret seg.
```

Sink :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 22
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print **Fact** TaintFlags WEB, XSS

```
 19|</head>
 20|<body>
 21|<% if (ok) { %>
>22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
 24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status  25|har endret seg.
```

Issue FFD1B39F92A685E3710F537D9B2CE9AC:

Source :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 24
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
 21|<% if (ok) { %>
 22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
>24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status
 25|har endret seg.
 26|<% } %>
 27|</body>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/endreSkjemastatus.jsp 24 **Action**
InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print **Fact** TaintFlags WEB, XSS

```
 21|<% if (ok) { %>
 22|Endret valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>.
 23|<% } else { %>
>24|Klarte ikke Ã¥ endre valideringstatus for skjema <%= request.getParameter("id") %> til status <%=
request.getParameter("status")%>, mest sannsynlig fordi gjeldende status
 25|har endret seg.
 26|<% } %>
 27|</body>
```

5.3 innsendinger.jsp

Issue 1556FD0990CAE7E7A9A3F2BE359C3B56:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 128
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
 125|<th><input value="<%=value%>" type="text" name="id" placeholder="eks: 1000-1200"></th>
 126|<% value = request.getParameter("fylke") == null || request.getParameter("fylke").equals("null") ? "" :
request.getParameter("fylke"); %>
 127|<th><input value="<%=value%>" type="text" name="fylke" placeholder="eks: 19"></th>
>128|<% value = request.getParameter("kommune") == null || request.getParameter("kommune").equals("null") ? "" :
request.getParameter("kommune"); %>
 129|<th><input value="<%=value%>" type="text" name="kommune" placeholder="eks: Enebakk,Ski"></th>
 130|<% value = request.getParameter("krets") == null || request.getParameter("krets").equals("null") ? "" :
request.getParameter("krets"); %>
 131|<th><input value="<%=value%>" type="text" name="krets" placeholder="eks: 0014"></th>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 129
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print **Fact** TaintFlags WEB, XSS

```
 126|<% value = request.getParameter("fylke") == null || request.getParameter("fylke").equals("null") ? "" :
request.getParameter("fylke"); %>
 127|<th><input value="<%=value%>" type="text" name="fylke" placeholder="eks: 19"></th>
 128|<% value = request.getParameter("kommune") == null || request.getParameter("kommune").equals("null") ? "" :
request.getParameter("kommune"); %>
>129|<th><input value="<%=value%>" type="text" name="kommune" placeholder="eks: Enebakk,Ski"></th>
 130|<% value = request.getParameter("krets") == null || request.getParameter("krets").equals("null") ? "" :
request.getParameter("krets"); %>
 131|<th><input value="<%=value%>" type="text" name="krets" placeholder="eks: 0014"></th>
 132|<% value = request.getParameter("os") == null || request.getParameter("os").equals("null") ? "" :
request.getParameter("os"); %>
```

Issue 38DE643602FED2815754A43AB6243483:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 124
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1
Fact Call Direct : javax.servlet.ServletRequest.getParameter

```
 121|<% String value = null; %>
 122|<% value = request.getParameter("status") == null || request.getParameter("status").equals("null") ? "" :
request.getParameter("status"); %>
 123|<th><input value="<%=value%>" type="text" name="status" placeholder="eks: !E,0" autofocus></th>
>124|<% value = request.getParameter("id") == null || request.getParameter("id").equals("null") ? "" :
request.getParameter("id"); %>
 125|<th><input value="<%=value%>" type="text" name="id" placeholder="eks: 1000-1200"></th>
 126|<% value = request.getParameter("fylke") == null || request.getParameter("fylke").equals("null") ? "" :
request.getParameter("fylke"); %>
 127|<th><input value="<%=value%>" type="text" name="fylke" placeholder="eks: 19"></th>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 125 **Action** InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags PRIMARY_KEY, WEB, XSS

```
 122|<% value = request.getParameter("status") == null || request.getParameter("status").equals("null") ? "" :
request.getParameter("status"); %>
 123|<th><input value="<%=value%>" type="text" name="status" placeholder="eks: !E,0" autofocus></th>
124|<% value = request.getParameter("id") == null || request.getParameter("id").equals("null") ? "" :
request.getParameter("id"); %>
>125|<th><input value="<%=value%>" type="text" name="id" placeholder="eks: 1000-1200"></th>
 126|<% value = request.getParameter("fylke") == null || request.getParameter("fylke").equals("null") ? "" :
request.getParameter("fylke"); %>
 127|<th><input value="<%=value%>" type="text" name="fylke" placeholder="eks: 19"></th>
 128|<% value = request.getParameter("kommune") == null || request.getParameter("kommune").equals("null") ? "" :
request.getParameter("kommune"); %>
```

Issue 3D0A8180A795944B7F60A3D2DDD686E8:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 134
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1
Fact Call Direct : javax.servlet.ServletRequest.getParameter

```
 131|<th><input value="<%=value%>" type="text" name="krets" placeholder="eks: 0014"></th>
 132|<% value = request.getParameter("os") == null || request.getParameter("os").equals("null") ? "" :
request.getParameter("os"); %>
 133|<th><input value="<%=value%>" type="text" name="os" placeholder="eks: 5-8"></th>
>134|<% value = request.getParameter("valinfo") == null || request.getParameter("valinfo").equals("null") ? "" :
request.getParameter("valinfo"); %>
 135|<th><input value="<%=value%>" type="text" name="valinfo" placeholder="eks: !"></th>
 136|<% value = request.getParameter("opprettet") == null || request.getParameter("opprettet").equals("null") ? "" :
request.getParameter("opprettet"); %>
 137|<th><input value="<%=value%>" type="text" name="opprettet" placeholder="eks: 22:3,22:4"></th>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 135
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 132|<% value = request.getParameter("os") == null || request.getParameter("os").equals("null") ? "" :
request.getParameter("os"); %>
 133|<th><input value="<%=value%>" type="text" name="os" placeholder="eks: 5-8"></th>
 134|<% value = request.getParameter("valinfo") == null || request.getParameter("valinfo").equals("null") ? "" :
request.getParameter("valinfo"); %>
>135|<th><input value="<%=value%>" type="text" name="valinfo" placeholder="eks: !"></th>
 136|<% value = request.getParameter("opprettet") == null || request.getParameter("opprettet").equals("null") ? "" :
request.getParameter("opprettet"); %>
 137|<th><input value="<%=value%>" type="text" name="opprettet" placeholder="eks: 22:3,22:4"></th>
138|<th></th>
```

Issue 6B2DB8E867626EF9F7510CE9CD778C25:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 132
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
 129|<th><input value="<%=value%>" type="text" name="kommune" placeholder="eks: Enebakk,Ski"></th>
 130|<% value = request.getParameter("krets") == null || request.getParameter("krets").equals("null") ? "" :
request.getParameter("krets"); %>
 131|<th><input value="<%=value%>" type="text" name="krets" placeholder="eks: 0014"></th>
>132|<% value = request.getParameter("os") == null || request.getParameter("os").equals("null") ? "" :
request.getParameter("os"); %>
 133|<th><input value="<%=value%>" type="text" name="os" placeholder="eks: 5-8"></th>
 134|<% value = request.getParameter("valinfo") == null || request.getParameter("valinfo").equals("null") ? "" :
request.getParameter("valinfo"); %>
 135|<th><input value="<%=value%>" type="text" name="valinfo" placeholder="eks: !"></th>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 133
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 130|<% value = request.getParameter("krets") == null || request.getParameter("krets").equals("null") ? "" :
request.getParameter("krets"); %>
 131|<th><input value="<%=value%>" type="text" name="krets" placeholder="eks: 0014"></th>
 132|<% value = request.getParameter("os") == null || request.getParameter("os").equals("null") ? "" :
request.getParameter("os"); %>
>133|<th><input value="<%=value%>" type="text" name="os" placeholder="eks: 5-8"></th>
 134|<% value = request.getParameter("valinfo") == null || request.getParameter("valinfo").equals("null") ? "" :
request.getParameter("valinfo"); %>
 135|<th><input value="<%=value%>" type="text" name="valinfo" placeholder="eks: !"></th>
 136|<% value = request.getParameter("opprettet") == null || request.getParameter("opprettet").equals("null") ? "" :
request.getParameter("opprettet"); %>
```

Issue 715BD52C9352DDE2F2E79F124C8091D0:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 130
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
 127|<th><input value="<%=value%>" type="text" name="fylke" placeholder="eks: 19"></th>
 128|<% value = request.getParameter("kommune") == null || request.getParameter("kommune").equals("null") ? "" :
request.getParameter("kommune"); %>
 129|<th><input value="<%=value%>" type="text" name="kommune" placeholder="eks: Enebakk,Ski"></th>
>130|<% value = request.getParameter("krets") == null || request.getParameter("krets").equals("null") ? "" :
request.getParameter("krets"); %>
 131|<th><input value="<%=value%>" type="text" name="krets" placeholder="eks: 0014"></th>
 132|<% value = request.getParameter("os") == null || request.getParameter("os").equals("null") ? "" :
request.getParameter("os"); %>
 133|<th><input value="<%=value%>" type="text" name="os" placeholder="eks: 5-8"></th>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 131
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 128|<% value = request.getParameter("kommune") == null || request.getParameter("kommune").equals("null") ? "" :
request.getParameter("kommune"); %>
 129|<th><input value="<%=value%>" type="text" name="kommune" placeholder="eks: Enebakk,Ski"></th>
 130|<% value = request.getParameter("krets") == null || request.getParameter("krets").equals("null") ? "" :
request.getParameter("krets"); %>
>131|<th><input value="<%=value%>" type="text" name="krets" placeholder="eks: 0014"></th>
 132|<% value = request.getParameter("os") == null || request.getParameter("os").equals("null") ? "" :
request.getParameter("os"); %>
 133|<th><input value="<%=value%>" type="text" name="os" placeholder="eks: 5-8"></th>
 134|<% value = request.getParameter("valinfo") == null || request.getParameter("valinfo").equals("null") ? "" :
request.getParameter("valinfo"); %>
```

Issue C84F1DA41EC5612B68884C72EA362EA0:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 126
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1
Fact Call Direct : javax.servlet.ServletRequest.getParameter

```
 123|<th><input value="<%=value%>" type="text" name="status" placeholder="eks: !E,O" autofocus></th>
124|<% value = request.getParameter("id") == null || request.getParameter("id").equals("null") ? "" :
request.getParameter("id"); %>
 125|<th><input value="<%=value%>" type="text" name="id" placeholder="eks: 1000-1200"></th>
>126|<% value = request.getParameter("fylke") == null || request.getParameter("fylke").equals("null") ? "" :
request.getParameter("fylke"); %>
 127|<th><input value="<%=value%>" type="text" name="fylke" placeholder="eks: 19"></th>
 128|<% value = request.getParameter("kommune") == null || request.getParameter("kommune").equals("null") ? "" :
request.getParameter("kommune"); %>
 129|<th><input value="<%=value%>" type="text" name="kommune" placeholder="eks: Enebakk,Ski"></th>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 127
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print **Fact** TaintFlags WEB, XSS

```
 124|<% value = request.getParameter("id") == null || request.getParameter("id").equals("null") ? "" :
request.getParameter("id"); %>
 125|<th><input value="<%=value%>" type="text" name="id" placeholder="eks: 1000-1200"></th>
 126|<% value = request.getParameter("fylke") == null || request.getParameter("fylke").equals("null") ? "" :
request.getParameter("fylke"); %>
>127|<th><input value="<%=value%>" type="text" name="fylke" placeholder="eks: 19"></th>
 128|<% value = request.getParameter("kommune") == null || request.getParameter("kommune").equals("null") ? "" :
request.getParameter("kommune"); %>
 129|<th><input value="<%=value%>" type="text" name="kommune" placeholder="eks: Enebakk,Ski"></th>
 130|<% value = request.getParameter("krets") == null || request.getParameter("krets").equals("null") ? "" :
request.getParameter("krets"); %>
```

Issue C8F7CB035DE712C3227038DB838EA5EE:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 122
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1
Fact Call Direct : javax.servlet.ServletRequest.getParameter

```
 119|<tbody>
 120|<tr>
 121|    <% String value = null; %>
>122|    <% value = request.getParameter("status") == null || request.getParameter("status").equals("null") ? "" :
request.getParameter("status"); %>
 123|    <th><input value="<%=value%>" type="text" name="status" placeholder="eks: !E,O" autofocus></th>
124|    <% value = request.getParameter("id") == null || request.getParameter("id").equals("null") ? "" :
request.getParameter("id"); %>
 125|    <th><input value="<%=value%>" type="text" name="id" placeholder="eks: 1000-1200"></th>
```

Sink :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 123 **Action** InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print **Fact** TaintFlags WEB, XSS

```
 120|<tr>
 121|    <% String value = null; %>
```

```
 122|    <% value = request.getParameter("status") == null || request.getParameter("status").equals("null") ? "" :
request.getParameter("status"); %>
>123|    <th><input value="<%=value%>" type="text" name="status" placeholder="eks: !E,O" autofocus></th>
124|    <% value = request.getParameter("id") == null || request.getParameter("id").equals("null") ? "" :
request.getParameter("id"); %>
 125|    <th><input value="<%=value%>" type="text" name="id" placeholder="eks: 1000-1200"></th>
 126|    <% value = request.getParameter("fylke") == null || request.getParameter("fylke").equals("null") ? "" :
request.getParameter("fylke"); %>
```

Issue D442208A6630804F3FBDE46AB8BE44E6:

Source :
File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 136
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
 133|<th><input value="<%=value%>" type="text" name="os" placeholder="eks: 5-8"></th>
 134|<% value = request.getParameter("valinfo") == null || request.getParameter("valinfo").equals("null") ? "" :
request.getParameter("valinfo"); %>
 135|<th><input value="<%=value%>" type="text" name="valinfo" placeholder="eks: !"></th>
>136|<% value = request.getParameter("opprettet") == null || request.getParameter("opprettet").equals("null") ? "" :
request.getParameter("opprettet"); %>
 137|<th><input value="<%=value%>" type="text" name="opprettet" placeholder="eks: 22:3,22:4"></th>
 138|<th></th>
 139|<th></th>
```

Sink :

File valg_overvaking/src/main/webapp/jsp/innsendinger.jsp 137
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 134|    <% value = request.getParameter("valinfo") == null || request.getParameter("valinfo").equals("null") ? "" :
request.getParameter("valinfo"); %>
 135|    <th><input value="<%=value%>" type="text" name="valinfo" placeholder="eks: !"></th>
 136|    <% value = request.getParameter("opprettet") == null || request.getParameter("opprettet").equals("null") ? ""
: request.getParameter("opprettet"); %>
>137|    <th><input value="<%=value%>" type="text" name="opprettet" placeholder="eks: 22:3,22:4"></th>
 138|    <th></th>
 139|    <th></th>
 140|</tr>
```

5.4 visKommune.jsp

Issue 09AE72E4C91361015D524465C6978143:

Source :
File valg_overvaking/src/main/webapp/jsp/visKommune.jsp 13
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
 10|String kommune = null;
 11|String kommentar = "";
 12|if (request.getParameter("kommune") != null) {
>13|    kommune = request.getParameter("kommune");
 14|    if (kommune.length() > 0) {
 15|        innsendinger = overvakingService.hentInnsendingKommune(kommune);  16|        if (innsendinger == null ||
innsendinger.size() == 0) {
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visKommune.jsp 65
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 62|<FORM NAME="form1" METHOD="POST">
 63|    <div id="tabs">
 64|        <ul>
>65|            <li><a href="#tab-innsending">Innsendinger (<%=kommune%>)</a></li>
 66|            <li><a href="#tab-kommune">Kommuneinfo</a></li>
67|        </ul>
 68|        <div id="tab-kommune">
```

5.5 visRapport.jsp

Issue 02A3FCD095C91D593F4EBE69070F5071:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 11
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
  9|<%
 10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
>11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|
String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|    String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 109
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 106|<TBODY>
 107|<TR>
 108|    <TD><label><INPUT TYPE="Text" VALUE="<%=rapportnr%>" name="rapportnr"></label></TD>
>109|    <TD><label><INPUT TYPE="Text" VALUE="<%=fylkenr%>" name="fylkenr"></label></TD>
 110|    <TD><label><INPUT TYPE="Text" VALUE="<%=kommnr%>" name="kommunenr"></label></TD>
 111|    <TD><label><INPUT TYPE="Text" VALUE="<%=kretsnr%>" name="kretsnr"></label></TD>
 112|    <TD><label><INPUT TYPE="Text" VALUE="<%=bydelsnr%>" name="bydelsnr"></label></TD>
```

Issue 50C14C7BA0915B7402D7A64695D58647:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 12
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
  9|<%
 10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
>12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";
 13|    String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|    String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";  15|
StringBuffer xmlFormatert = new StringBuffer("");
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 110
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print **Fact** TaintFlags WEB, XSS

```
 107|<TR>
 108|    <TD><label><INPUT TYPE="Text" VALUE="<%=rapportnr%>" name="rapportnr"></label></TD>
 109|    <TD><label><INPUT TYPE="Text" VALUE="<%=fylkenr%>" name="fylkenr"></label></TD>
>110|    <TD><label><INPUT TYPE="Text" VALUE="<%=kommnr%>" name="kommunenr"></label></TD>
 111|    <TD><label><INPUT TYPE="Text" VALUE="<%=kretsnr%>" name="kretsnr"></label></TD>
 112|    <TD><label><INPUT TYPE="Text" VALUE="<%=bydelsnr%>" name="bydelsnr"></label></TD>  113|</TR>
```

Issue A12ED788E0FF7D436ABC0C803F1521B4:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 14
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
 11|String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|String
kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
>14|String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
 15|StringBuffer xmlFormatert = new StringBuffer("");
 16|if (rapportnr != null && rapportnr.length() > 0) {
 17|    String rapporturl = "/" + rapportnr;
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 112
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 109|        <TD><label><INPUT TYPE="Text" VALUE="<%=fylkenr%>" name="fylkenr"></label></TD>
 110|        <TD><label><INPUT TYPE="Text" VALUE="<%=kommnr%>" name="kommunenr"></label></TD>
 111|        <TD><label><INPUT TYPE="Text" VALUE="<%=kretsnr%>" name="kretsnr"></label></TD>
>112|        <TD><label><INPUT TYPE="Text" VALUE="<%=bydelsnr%>" name="bydelsnr"></label></TD>
 113|    </TR>
 114|    </TBODY>
 115|</TABLE>
```

Issue D1BCEF795D8D01D129A2F3557383A8E2:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 10
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
  7|<%@ include file="/jspf/top.jspf" %>
  8|
  9|<%
>10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|
String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 108
Action InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print
Fact TaintFlags WEB, XSS

```
 105|</THEAD>
 106|<TBODY>
 107|<TR>
>108|    <TD><label><INPUT TYPE="Text" VALUE="<%=rapportnr%>" name="rapportnr"></label></TD>
 109|    <TD><label><INPUT TYPE="Text" VALUE="<%=fylkenr%>" name="fylkenr"></label></TD>
 110|    <TD><label><INPUT TYPE="Text" VALUE="<%=kommnr%>" name="kommunenr"></label></TD>
 111|    <TD><label><INPUT TYPE="Text" VALUE="<%=kretsnr%>" name="kretsnr"></label></TD>
```

Issue F809D6531F8CFFBFECBFFB0615812A6F:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 13
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
 10|String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";
>13|String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
 15|StringBuffer xmlFormatert = new StringBuffer("");
 16|if (rapportnr != null && rapportnr.length() > 0) {
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 111 **Action**
InCall print(0)
Rule id 45BF957F-1A34-4E28-9B34-FEB83EC96792
Fact Call Direct : javax.servlet.jsp.JspWriter.print **Fact** TaintFlags
WEB, XSS

```
 108|    <TD><label><INPUT TYPE="Text" VALUE="<%=rapportnr%>" name="rapportnr"></label></TD>
 109|    <TD><label><INPUT TYPE="Text" VALUE="<%=fylkenr%>" name="fylkenr"></label></TD>
 110|    <TD><label><INPUT TYPE="Text" VALUE="<%=kommnr%>" name="kommunenr"></label></TD>
>111|    <TD><label><INPUT TYPE="Text" VALUE="<%=kretsnr%>" name="kretsnr"></label></TD>
 112|    <TD><label><INPUT TYPE="Text" VALUE="<%=bydelsnr%>" name="bydelsnr"></label></TD>
 113|</TR>
 114|</TBODY>
```

[High] Race Condition / Format Flaw

Time and State

Analysis: Reliability Issue

Abstract

The methods `parse()` and `format()` in `java.text.Format` contain a design flaw that can cause one user to see another user's data.

Explanation

The methods `parse()` and `format()` in `java.text.Format` contains a race condition that can cause one user to see another user's data.

Example 1: The code below shows how this design flaw can manifest itself.

```
public class Common {
     private static SimpleDateFormat
dateFormat;     ...      public String
format(Date date) {         return
dateFormat.format(date);
    }     ...      final OtherClass
dateFormatAccess=new OtherClass();     ...
    public void function_running_in_thread1(){
        System.out.println("Time  in  thread  1  should  be  12/31/69  4:00  PM,  found:  "+  dateFormatAccess.format(new
Date(0)));
    }
    public void function_running_in_thread2(){
        System.out.println("Time in thread 2 should be around 12/29/09 6:26 AM, found: "+ dateFormatAccess.format(new
Date(System.currentTimeMillis())));
    }
}
```

While this code will behave correctly in a single-user environment, if two threads run it at the same time they could produce the following output:

Time in thread 1 should be 12/31/69 4:00 PM, found: 12/31/69 4:00 PM Time in thread 2 should be around 12/29/09 6:26 AM, found: 12/31/69 4:00 PM

In this case, the date from the first thread is shown in the output from the second thread due a race condition in the implementation of `format()`.

Recommendation

Use synchronization to protect against race conditions whenever `parse()` and `format()` in class `java.text.Format`.

Example 2: The code below shows two ways that the code from Example 1 can be corrected using synchronization constructs.

```
public class Common {
     private static SimpleDateFormat dateFormat;
...      public synchronized String format1(Date
date) {         return dateFormat.format(date);
    }
    public String format2(Date date) {
synchronized(dateFormat)
        {
            return dateFormat.format(date);
        }
    }
}
```

Alternatively, use `org.apache.commons.lang.time.FastDateFormat` class, which is a threadsafe version of `java.text.SimpleDateFormat`.

5.6 Format.java

Issue 2C4E871E3EE0428FE5F2855AFC1CDA3C:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/rapport/Format.java 53
Fact Name: java.text.NumberFormat.format

```
 52|static String pst(float f) {
>53|    return prosFmt.format(HUNDRE * f);
```

Issue 32CAA5D5F3AA98F24A7394BD1B83FDA7:

Sink :

File valg_backend/src/main/java/no/valg/valgnatt/backend/rapport/Format.java 59
Fact Name: java.text.NumberFormat.format

```
 58|static String pstNN(Float f) {
>59|    return f == null ? null : prosFmt.format(HUNDRE * f);
```

Issue 3C950AA23B11B3425A8C80CDEE822FB8:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/rapport/Format.java 95
Fact Name: java.text.NumberFormat.format

```
 94|public static String kvotient(double kvot) {
>95|    return kvotFmt.format(kvot);
```

Issue 488026112191F0D5692E2B4E16A30CB5:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/rapport/Format.java 69
Fact Name: java.text.Format.format

```
 68|static String pstNN(int teller, int nevner) {
>69|    return nevner == 0 ? null : prosFmt.format(new BigDecimal(HUNDRE * teller).divide(new BigDecimal(nevner), 1,
BigDecimal.ROUND_HALF_UP));
```

Issue 95E63A68EEA8FE613558F5169D103F20:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/rapport/Format.java 90
Fact Name: java.text.Format.format

```
 88|    BigDecimal b1 = new BigDecimal(HUNDRE * teller1).divide(new BigDecimal(nevner1), TI,
BigDecimal.ROUND_HALF_UP);
 89|    BigDecimal b2 = new BigDecimal(HUNDRE * teller2).divide(new BigDecimal(nevner2), TI,
BigDecimal.ROUND_HALF_UP);
>90|    return diffprosFmt.format(b1.subtract(b2));
```

Issue D9913D3F5E34699306DC6566742D121F:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/rapport/Format.java 47
Fact Name: java.text.NumberFormat.format

```
 44| **/
 45|
 46|static String enDesimal(double d) {
>47|    return prosFmt.format(d);
```

Issue DD0B1770EF97195AF2B7B98E570841D6:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/rapport/Format.java 78
Fact Name: java.text.NumberFormat.format

```
 77|static String pstEndringNN(Float f1, Float f2) {
>78|    return f1 == null || f2 == null ? null : diffprosFmt.format(HUNDRE * (f1 - f2));
```

5.7 StemmeInnsender.java

Issue 247451A1DD0CB5F3DEA70758FB7D2EFC:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 242 **Fact** Name: java.text.DateFormat.parse

```
 240|private long tms(String filnavnTidspunkt) throws ParseException {
 241|    if (filnavnTidspunkt.length() == 18) {
>242|        return SDF18.parse(filnavnTidspunkt).getTime();
 243|    } else {
 244|        return SDF15.parse(filnavnTidspunkt).getTime();
```

Issue 9F3623C10C23FD0E9980C85236A5E320:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 244
Fact Name: java.text.DateFormat.parse

```
 241|    if (filnavnTidspunkt.length() == 18) {
 242|        return SDF18.parse(filnavnTidspunkt).getTime();
 243|    } else {
>244|        return SDF15.parse(filnavnTidspunkt).getTime();
```

5.8 Stemmeskjema2Filsystem.java

Issue 714C58E6970A5DBF401C4EDCC66AC7D8:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/Stemmeskjema2Filsystem.java 74
Fact Name: java.text.DateFormat.format

```
 71|    LOG.log(Level.INFO, "Katalogen " + f.getAbsolutePath() + " finnes ikke, mÃ¥ lages");
72|    f.mkdirs();
 73|}
>74|File file = new File(f.getAbsoluteFile() + File.separator + sdf3.format(opprettet) + "-" + kretsnr + ".json");
75|if (!file.exists()) {
 76|    LOG.log(Level.INFO, "File " + file.getAbsoluteFile() + " finnes ikke, mÃ¥ lages");
 77|    BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file), "UTF-8"));
```

5.9 StemmeskjemaInfo.java

Issue 4862E2C6247B0283F60BCD19F707F28C:

Sink :
File valg_overvaking/src/main/java/no/valg/valgnatt/overvaking/StemmeskjemaInfo.java 14
Fact Name: java.text.DateFormat.format

```
 11|private Integer antBehandles = 0;
 12|private List<String> sisteInnsendinger = new ArrayList<>();
 13|private SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss, dd.MMM"); >14|private String timestamp = 
sdf.format(new Date());
 15|
 16|public Integer getAntTotalt() {
 17|    return antTotalt;
```

[Medium] Denial of Service / StringBuilder

Input Validation and Representation

Analysis: Not an Issue / Reliability Issue

Abstract

Appending untrusted data to `StringBuilder` instance initialized with the default backing-array size can cause the JVM to over-consume heap memory space.

Explanation

Appending user-controlled data to a `StringBuilder` instance initialized with the default backing character array size (16) can cause the application to consume large amounts of heap memory while resizing the underlying array to fit user's data. Everytime new data is appended to a `StringBuilder` instance, it will try to fit it on its backing character array. If data does not fit, a new array will be created doubling the previous size while the old array will remain in the heap until it is garbage collected. This defect can be used to execute a Denial of Service (DoS) attack.

Example 1: User-controlled data is appended to a `StringBuilder` instance initialized with the default constructor.

```
   ...
   StringBuilder sb = new StringBuilder();
sb.append(request.getParameter("foo"));     ...
```

Recommendation

Initialize the `StringBuilder` with a size similar to the length of the expected data in order to reduce the number of times the backing array needs to be resized. Check the size of the data before appending it to a `StringBuilder` instance. **Example 2:** User-controlled data is appended to a `StringBuilder` instance initialized with the default constructor.

```
    ...     private final int
MAX_DATA = 128;
    private final int EXPECTED_BUFFER_DATA = 1024;
    StringBuilder sb = new StringBuilder(EXPECTED_BUFFER_DATA);
...
    String data = request.getParameter("foo");
if (data.length() < MAX_DATA) sb.append(data);
...
```

5.10 MainMottak.java

Issue 742B7CEBCA16E8E7A6BC6997D7A75BAB:

Source :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 100
Action OutCall readLine(return)
Rule id 2A983818-6144-4E5A-BE5C-E955BF9A3FBA
Fact Call Direct : java.io.BufferedReader.readLine

```
  97|BufferedReader br = new BufferedReader(new InputStreamReader(is));
  98|String line;
  99|StringBuilder sb = new StringBuilder();
>100|while ((line = br.readLine()) != null) {
 101|    sb.append(line).append('\n');
 102|}
 103|br.close();
```

Sink :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 101 **Action** InCall append(0)
Rule id F2BD85B8-504E-4D52-967C-E00A043BAFAD
Fact Call Direct : java.lang.StringBuilder.append
Fact TaintFlags NO_NEW_LINE, STREAM

```
  98|String line;
  99|StringBuilder sb = new StringBuilder();
 100|while ((line = br.readLine()) != null) {
>101|    sb.append(line).append('\n');
 102|}
 103|br.close();
 104|result = sb.toString();
```

Issue 8C416761E928E7F5BC301A8705AC0537:

Source :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 95
Action OutCall getInputStream(return)
Rule id E080F8D1-F512-49DA-B076-CBA2A217C538
Fact Call Direct : java.net.URLConnection.getInputStream

```
 92|responseMessage = conn.getResponseMessage();
 93|InputStream is = conn.getErrorStream();
 94|if (is == null) {
>95|    is = conn.getInputStream();
 96|}
 97|BufferedReader br = new BufferedReader(new InputStreamReader(is));  98|String line;
```

Sink :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 101
Action InCall append(0)
Rule id F2BD85B8-504E-4D52-967C-E00A043BAFAD
Fact Call Direct : java.lang.StringBuilder.append
Fact TaintFlags NO_NEW_LINE, WEBSERVICE, XSS

```
  98|String line;
  99|StringBuilder sb = new StringBuilder();
 100|while ((line = br.readLine()) != null) {
>101|    sb.append(line).append('\n');
 102|}
 103|br.close();
 104|result = sb.toString();
```

5.11 MottakFraBackendServlet.java

Issue 2163DD8AF67A6084E946FD9555664E1F:

Source :
File valg_frontend/src/main/java/no/valg/valgnatt/frontend/MottakFraBackendServlet.java 43
Action OutCall getReader(return)

Rule id 23311ABB-0992-43C8-91A2-22F474402E63
Fact Call Direct : javax.servlet.ServletRequest.getReader

```
 40|public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {  41|
try {
 42|        StringBuilder sb = new StringBuilder();
>43|        BufferedReader br = req.getReader();
 44|        String line;
 45|        while ((line = br.readLine()) != null) {
 46|            sb.append(line).append('\n');
```

Sink :
File valg_frontend/src/main/java/no/valg/valgnatt/frontend/MottakFraBackendServlet.java 46
Action InCall append(0)
Rule id F2BD85B8-504E-4D52-967C-E00A043BAFAD
Fact Call Direct : java.lang.StringBuilder.append
Fact TaintFlags NO_NEW_LINE, WEB, XSS

```
 43|BufferedReader br = req.getReader();
 44|String line;
 45|while ((line = br.readLine()) != null) {
>46|    sb.append(line).append('\n');
 47|}
 48|br.close();
```

Issue 7E52AB23F4B4D29AD11E9AE0D7BC46F1:

Source :
File valg_frontend/src/main/java/no/valg/valgnatt/frontend/MottakFraBackendServlet.java 45
Action OutCall readLine(return)
Rule id 2A983818-6144-4E5A-BE5C-E955BF9A3FBA
Fact Call Direct : java.io.BufferedReader.readLine

```
 42|StringBuilder sb = new StringBuilder();
 43|BufferedReader br = req.getReader();
 44|String line;
>45|while ((line = br.readLine()) != null) {
 46|    sb.append(line).append('\n');
 47|}
 48|br.close();
```

Sink :
File valg_frontend/src/main/java/no/valg/valgnatt/frontend/MottakFraBackendServlet.java 46
Action InCall append(0)
Rule id F2BD85B8-504E-4D52-967C-E00A043BAFAD
Fact Call Direct : java.lang.StringBuilder.append
Fact TaintFlags NO_NEW_LINE, STREAM

```
 43|BufferedReader br = req.getReader();
 44|String line;
 45|while ((line = br.readLine()) != null) {
>46|    sb.append(line).append('\n');
 47|}
 48|br.close();
```

5.12 StemmeInnsender.java

Issue FED1599C1EF8044DD260709BEB81F608:

Source :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 168
Action OutCall readLine(return)
Rule id 2A983818-6144-4E5A-BE5C-E955BF9A3FBA
Fact Call Direct : java.io.BufferedReader.readLine

```
 165|BufferedReader br = new BufferedReader(new InputStreamReader(is));
 166|StringBuilder content = new StringBuilder();
 167|String line;
>168|while ((line = br.readLine()) != null) {
 169|    content.append(line);
 170|}
 171|br.close();
```

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 169
Action InCall append(0)
Rule id F2BD85B8-504E-4D52-967C-E00A043BAFAD
Fact Call Direct : java.lang.StringBuilder.append
Fact TaintFlags NO_NEW_LINE, STREAM

```
 166|StringBuilder content = new StringBuilder();
```

```
String line;
while ((line = br.readLine()) != null) {
    content.append(line);
}
br.close();
long waitedMs = 0;
```

[Medium] J2EE Misconfiguration / Missing Data Transport Constraint

Environment

Analysis: Suspicious

Abstract

A security constraint that does not specify a user data constraint cannot guarantee that restricted resources will be protected at the transport layer.

Explanation

`web.xml` security constraints are typically used for role based access control, but the optional `userdata-constraint` element specifies a transport guarantee that prevents content from being transmitted insecurely.

Within the `<user-data-constraint>` tag, the `<transport-guarantee>` tag defines how communication should be handled. There are three levels of transport guarantee:

1) `NONE` means that the application does not require any transport guarantees. 2) `INTEGRAL` means that the application requires that data sent between the client and server be sent in such a way that it cannot be changed in transit. 3) `CONFIDENTIAL` means that the application requires that data be transmitted in a fashion that prevents other entities from observing the contents of the transmission.

In most circumstances, the use of `INTEGRAL` or `CONFIDENTIAL` means that SSL/TLS is required. If the `<user-data-constraint>` and `<transport-guarantee>` tags are omitted, the transport guarantee defaults to `NONE`.

Example 1: The following security constraint does not specify a transport guarantee.

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Storefront</web-resource-name>
        <description>Allow Customers and Employees access to online store front</description>
        <url-pattern>/store/shop/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description>Anyone</description>
        <role-name>anyone</role-name>
    </auth-constraint>
</security-constraint>
```

This category was derived from the Cigital Java Rulepack. http://www.cigital.com/

Recommendation

Specify a `CONFIDENTIAL` transport guarantee whenever you define a authorization constraint. Once you decide to encrypt traffic to any part of your application, do not make the mistake of allowing unencrypted traffic to other parts of the application, which could allow session cookies or other sensitive information to be transmitted over insecure channels.

Example 2: The following security constraint specifies a `CONFIDENTIAL` transport guarantee.

```
<security-constraint>
```

```
    <web-resource-collection>
        <web-resource-name>Storefront</web-resource-name>
        <description>Allow Customers and Employees access to online store front</description>
        <url-pattern>/store/shop/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description>Anyone</description>
        <role-name>anyone</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint> </security-constraint>
```

5.13 web.xml

Issue 4745492F4C56918DB027E1B7D27069F8:

Sink :
File valg_frontend/src/main/webapp/WEB-INF/web.xml 83

```
 80|    <auth-method>BASIC</auth-method>
 81|</login-config>
 82|
>83|<security-constraint>
 84|    <web-resource-collection>
 85|        <web-resource-name>Admin</web-resource-name>
 86|        <url-pattern>/valg_frontend</url-pattern>
```

Issue 4745492F4C56918DB027E1B7D27069F9:

Sink :
File valg_overvaking/src/main/webapp/WEB-INF/web.xml 74

```
 71|    </form-login-config>
 72|</login-config>
 73|
>74|<security-constraint>
 75|    <web-resource-collection>
 76|        <web-resource-name>Admin</web-resource-name>
 77|        <url-pattern>/jsp/diverse.jsp</url-pattern>
```

[Medium] J2EE Misconfiguration / Missing Error Handling

Environment

Analysis: Suspicious

Abstract

A web application must define default error pages in order to prevent attackers from mining information from the application container's built-in error response.

Explanation

When an attacker explores a web site looking for vulnerabilities, the amount of information that the site provides is crucial to the eventual success or failure of any attempted attacks. If the application shows the attacker a stack trace, it relinquishes information that makes the attacker's job significantly easier. For example, a stack trace might show the attacker a malformed SQL query string, the type of database being used, and the version of the application container. This information enables the attacker to target known vulnerabilities in these components.

The application configuration should specify a default error page in order to guarantee that the application will never leak error messages to an attacker. Handling standard HTTP error codes is useful and user-friendly in addition to being a good security practice, and a good configuration will also define a last-chance error handler that catches any exception that could possibly be thrown by the application.

Recommendation

A web application must be configured with a default error page. Your `web.xml` should include at least the following entries:

```
<error-page>
   <exception-type>java.lang.Throwable</exception-type>
<location>/error.jsp</location>
</error-page>
<error-page>
   <error-code>404</error-code> <location>/error.jsp</location>
</error-page>
<error-page>
   <error-code>500</error-code> <location>/error.jsp</location>
</error-page>
```

5.14 web.xml

Issue 399A248E35AE0FBB04255DE45FA9754C:

Sink :
File valg_backend/src/main/webapp/WEB-INF/web.xml 5
Action /web-app

```
 2|xmlns="http://java.sun.com/xml/ns/javaee"
 3|xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  4|xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
>5|http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 6|
 7|<display-name>valg_backend</display-name>
```

Issue 399A248E35AE0FBB04255DE45FA9754D:

Sink :
File valg_frontend/src/main/webapp/WEB-INF/web.xml 5
Action /web-app

```
 2|xmlns="http://java.sun.com/xml/ns/javaee"
 3|xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  4|xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
>5|http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 6|
 7|<display-name>valg_frontend</display-name>
```

Issue 399A248E35AE0FBB04255DE45FA9754E:

Sink :
File valg_mottak/src/main/webapp/WEB-INF/web.xml 5
Action /web-app

```
 2|xmlns="http://java.sun.com/xml/ns/javaee"
 3|xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  4|xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
>5|http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 6|
 7|<display-name>valg_mottak</display-name>
```

[Medium] Missing Check against Null

API Abuse

Analysis: Reliability Issue

Abstract

The program can dereference a null pointer because it does not check the return value of a function that might return null.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break.

Two dubious assumptions that are easy to spot in code are "this function call can never fail" and "it doesn't matter if this function call fails". When a programmer ignores the return value from a function, they implicitly state that they are operating under one of these assumptions.

Example 1: The following code does not check to see if the string returned by `getParameter()` is `null` before calling the member function `compareTo()`, potentially causing a null dereference.

```
String itemName = request.getParameter(ITEM_NAME);
if (itemName.compareTo(IMPORTANT_ITEM)) { ... }
...
```

Example 2:. The following code shows a system property that is set to `null` and later dereferenced by a programmer who mistakenly assumes it will always be defined.

```
System.clearProperty("os.name");
...
String os = System.getProperty("os.name");
if (os.equalsIgnoreCase("Windows 95") )
System.out.println("Not supported");
```

The traditional defense of this coding error is:

"I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or simply allow the program to die dereferencing a null value."

But attackers are skilled at finding unexpected paths through programs, particularly when exceptions are involved.

Recommendation

If a function can return an error code or any other evidence of its success or failure, always check for the error condition, even if there is no obvious way for it to occur. In addition to preventing security errors, many initially mysterious bugs have eventually led back to a failed method call with an unchecked return value.

Create an easy to use and standard way for dealing with failure in your application. If error handling is straightforward, programmers will be less inclined to omit it. One approach to standardized error handling is to write wrappers around commonly-used functions that check and handle error conditions without additional programmer intervention. When wrappers are implemented and adopted, the use of non-wrapped equivalents can be prohibited and enforced by using custom rules.

Example 3: The following code implements a wrapper around `getParameter()` that checks the return value of `getParameter()` against `null` and uses a default value if the requested parameter is not defined.

```
String safeGetParameter (HttpRequest request, String name)
{
    String  value  =  request.getParameter(name);
if (value == null) {
        return getDefaultValue(name)
    }
    return value;
}
```

5.15 LoadSave.java

Issue 5A6B93E58E383C312A696B238D38374A:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/forberedvalg/LoadSave.java 41

Action Assign getClassLoader() : Class.getClassLoader may return NULL **Rule id** 4280F38B-9FDB-454E-B495-89CF45CD51B7

```
 38|LOG.info("Leser " + fullName);
 39|
 40|Reader reader;
>41|InputStream resourceAsStream = LoadSave.class.getClassLoader().getResourceAsStream(fullName);
 42|if (resourceAsStream != null) {  43|     reader = new InputStreamReader(resourceAsStream);  44|} else {
```

Sink :
File valg_database/src/main/java/no/valg/valgnatt/database/forberedvalg/LoadSave.java 41
Action Assign getClassLoader() : Class.getClassLoader may return NULL **Rule id** 4280F38B-9FDB-454E-B495-89CF45CD51B7

```
 38|LOG.info("Leser " + fullName);
 39|
 40|Reader reader;
>41|InputStream resourceAsStream = LoadSave.class.getClassLoader().getResourceAsStream(fullName);
 42|if (resourceAsStream != null) {  43|     reader = new InputStreamReader(resourceAsStream);  44|} else {
```

5.16 V20150000_002__Baseline_Load.java

Issue 5FBC0B6EA6482318469A394FF5E53291:

Source :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 49
Action BranchTaken Branch taken: (resource~iterator < resource~iterator~len)

```
 47|BaseConnection base = getBaseConnection(connection);
 48|CopyManager cm = new CopyManager(base);
>49|for (Resource resource : resources) {
 50|    BufferedReader br = new BufferedReader(new InputStreamReader(resource.getInputStream()));
 51|    String colNames = br.readLine().replace(';', ',');
52|    br.close();
```

Sink :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 51
Action Assign readLine() : BufferedReader.readLine may return NULL
Rule id 4280F38B-9FDB-454E-B495-89CF45CD51B7

```
 48|CopyManager cm = new CopyManager(base);
 49|for (Resource resource : resources) {
 50|    BufferedReader br = new BufferedReader(new InputStreamReader(resource.getInputStream()));
>51|    String colNames = br.readLine().replace(';', ',');
 52|    br.close();
 53|    InputStream is = resource.getInputStream();
 54|    String tableName = resource.getFilename().substring(0, resource.getFilename().indexOf('.'));
```

5.17 visKommune.jsp

Issue 15736988E8A5A2AE2726509ADDE4FDB1:

Source :
File valg_overvaking/src/main/webapp/jsp/visKommune.jsp 12
Action BranchTaken Branch taken: (request.getParameter("kommune") != null)

```
  9|String kommnavn = null;
 10|String kommune = null;
 11|String kommentar = "";
>12|if (request.getParameter("kommune") != null) {
 13|    kommune = request.getParameter("kommune");
 14|    if (kommune.length() > 0) {
 15|        innsendinger = overvakingService.hentInnsendingKommune(kommune);
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visKommune.jsp 13
Action Assign kommune = getParameter(...) : ServletRequest.getParameter may return NULL
Rule id 4280F38B-9FDB-454E-B495-89CF45CD51B7

```
 10|String kommune = null;
 11|String kommentar = "";
 12|if (request.getParameter("kommune") != null) {
>13|    kommune = request.getParameter("kommune");
 14|    if (kommune.length() > 0) {
 15|        innsendinger = overvakingService.hentInnsendingKommune(kommune);  16|        if (innsendinger == null ||
innsendinger.size() == 0) {
```

5.18 visStemmeskjema.jsp

Issue C58153D4B4201E9A53F7A39D14F5A752:

Source :
File valg_overvaking/src/main/webapp/jsp/visStemmeskjema.jsp 11
Action BranchTaken Branch taken: (request.getParameter("id") != null)

```
  9|<%
 10|    String data = "";
>11|    if (request.getParameter("id") != null) {
 12|        String id = request.getParameter("id");
 13|        if (id.length() > 0) {
 14|            Innsending innsending = overvakingService.hentInnsending(Integer.valueOf(id));
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visStemmeskjema.jsp 12
Action Assign id = getParameter(...) : ServletRequest.getParameter may return NULL
Rule id 4280F38B-9FDB-454E-B495-89CF45CD51B7

```
  9|<%
 10|    String data = "";
 11|    if (request.getParameter("id") != null) {
>12|        String id = request.getParameter("id");
 13|        if (id.length() > 0) {
 14|            Innsending innsending = overvakingService.hentInnsending(Integer.valueOf(id));
 15|            if (innsending != null) {
```

[Medium] Missing Check for Null Parameter

API Abuse

Analysis: Reliability Issue

Abstract

This function violates the contract that it must compare its parameter with null.

Explanation

The Java standard requires that implementations of `Object.equals()`, `Comparable.compareTo()`, and `Comparator.compare()` must return a specified value if their parameters are null. Failing to follow this contract may result in unexpected behavior.

Example 1: The following implementation of the `equals()` method does not compare its parameter with null.

```
public boolean equals(Object object)
{    return
(toString().equals(object.toString())); }
```

Recommendation

Always follow the contract that `Object.equals()` must return `null` if it receives a null parameter.

Example 2: The previous example is rewritten to explicitly check for a null argument and return false if one is found.

```
public boolean equals(Object object)
{     if (object ==
null)         return
false;
    return (toString().equals(object.toString()));
}
```

5.19 ValgTypeAar.java

Issue 15E2FD56510C53061E26BE9AB14B5A9C:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/domene/ValgTypeAar.java 33
Action o = #param(0)
Rule id 1834C857-2848-41F2-98F7-D5C5F157E729

```
 32|@Override
>33|public boolean equals(Object o) {
 34|    return o != null && o.getClass() == ValgTypeAar.class || o.toString().equals(toString());
```

Sink :
File valg_database/src/main/java/no/valg/valgnatt/database/domene/ValgTypeAar.java 33
Action o = #param(0)
Rule id 1834C857-2848-41F2-98F7-D5C5F157E729

```
 32|@Override
>33|public boolean equals(Object o) {
 34|    return o != null && o.getClass() == ValgTypeAar.class || o.toString().equals(toString());
```

[Medium] Password Management / Empty Password in Configuration File

Environment

Analysis: Suspicious

Abstract

Using an empty string as a password is insecure.

Explanation

It is never appropriate to use an empty string as a password. It is too easy to guess.

Recommendation

Require that sufficiently hard-to-guess passwords protect all accounts and system resources. Consult the following references for help in establishing appropriate password guidelines.

5.20 db-servere.properties

Issue 226E3536A603F66BF51EE56684681449:

Sink :
File valg_utils/src/main/resources/db-servere.properties 40
Action db.stg.password

```
 37|# Sett opp en tunnel til stage-db:5432 pÃ¥ localhost:9998
 38|db.stg.url = jdbc:postgresql://localhost:9998/resultat
 39|db.stg.user = resultat
>40|db.stg.password =
 41|
 42|db.prodslave.driver = org.postgresql.Driver
 43|# Sett opp en tunnel til stage-db:5432 pÃ¥ localhost:9998
```

Issue C7534DE9962EB8B398BE39D6AD735FA3:

Sink :
File valg_utils/src/main/resources/db-servere.properties 46
Action db.prodslave.password

```
 43|# Sett opp en tunnel til stage-db:5432 pÃ¥ localhost:9998
 44|db.prodslave.url = jdbc:postgresql://localhost:7777/resultat
 45|db.prodslave.user = ro_bruker
>46|db.prodslave.password=
 47|db.localhost2017.driver=org.postgresql.Driver
 48|db.localhost2017.url=jdbc:postgresql://localhost:5432/valgnatt_2017  49|db.localhost2017.user=valgnatt
```

[Medium] Password Management / Password in Configuration File

Environment

Analysis: Not an Issue / Suspicious

Abstract

Storing a plaintext password in a configuration file may result in a system compromise.

Explanation

Storing a plaintext password in a configuration file allows anyone who can read the file access to the password-protected resource. Developers sometimes believe that they cannot defend the application from someone who has access to the configuration, but this attitude makes an attacker's job easier. Good password management guidelines require that a password never be stored in plaintext.

Recommendation

A password should never be stored in plaintext. Instead, the password should be entered by an administrator when the system starts. If that approach is impractical, a less secure but often adequate solution is to obfuscate the password and scatter the de-obfuscation material around the system so that an attacker has to obtain and correctly combine multiple system resources to decipher the password.

Some third-party products claim the ability to manage passwords in a more secure way. For example, WebSphere Application Server 4.x uses a simple XOR encryption algorithm for obfuscating values, but be skeptical about such facilities. WebSphere and other application servers offer outdated and relatively weak encryption mechanisms that are insufficient for security-sensitive environments. For a secure solution the only viable option is a proprietary one.

5.21 db-servere.properties

Issue 0AD0BB1D9A992C3E35A0CBB0C7F74646:

Sink :
File valg_utils/src/main/resources/db-servere.properties 50
Action db.localhost2017.password

```
 47|db.localhost2017.driver=org.postgresql.Driver
 48|db.localhost2017.url=jdbc:postgresql://localhost:5432/valgnatt_2017
 49|db.localhost2017.user=valgnatt
>50|db.localhost2017.password=valgnatt
```

[Medium] Poor Error Handling / Throw Inside Finally

Errors

Analysis: Bad Practice

Abstract

Using a `throw` statement inside a `finally` block breaks the logical progression through the `trycatch-finally`.

Explanation

In Java, `finally` blocks are always executed after their corresponding `try-catch` blocks and are often used to free allocated resources, such as file handles or database cursors. Throwing an exception in a `finally` block can bypass critical cleanup code since normal program execution will be disrupted.

Example 1: In the following code, the call to `stmt.close()` is bypassed when the `FileNotFoundException` is thrown.

```
public void processTransaction(Connection conn) throws FileNotFoundException
{
    FileInputStream fis =
null;     Statement stmt =
null;     try     {
        stmt = conn.createStatement();
        fis = new
FileInputStream("badFile.txt");        ...
}
    catch (FileNotFoundException fe)
    {
        log("File not found.");
    }     catch
(SQLException se)
    {
        //handle error
    }     finally     {
if (fis == null)
        {
            throw new FileNotFoundException();
}


        if (stmt != null)
        {
try
{
                stmt.close();
            }
            catch (SQLException e)
            {
                log(e);
            }
        }
    }
}
```

This category is from the Cigital Java Rulepack. http://www.cigital.com/

Recommendation

Never throw exceptions from within `finally` blocks. If you must re-throw an exception, do it inside a `catch` block so as not to interrupt the normal execution of the `finally` block. **Example 2:** The following code re-throws the `FileNotFoundException` in the `catch` block.

```
public void processTransaction(Connection conn) throws FileNotFoundException
{
    FileInputStream fis =
null;     Statement stmt =
null;     try     {
        stmt = conn.createStatement();
        fis = new
FileInputStream("badFile.txt");         ...
}
    catch (FileNotFoundException fe)
    {
        log("File not found.");
throw fe;
    }
    catch (SQLException se)
    {
        //handle error
    }
finally
{
        if (fis != null)
        {
try
{
                fis.close();
            }
            catch (IOException ie)
            {
                log(ie);
            }
        }
        if (stmt != null)
        {
try
{
                stmt.close();
            }             catch
(SQLException e)
            {
                log(e);
            }
        }
    }
}
```

5.22 EksportSkjema.java

Issue A229450F973DAF426BBBE570D735A3A6:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/eksport/EksportSkjema.java 36

```
 35|void skrivTilFil() throws IOException {
>36|    try (FileWriter fw = new FileWriter(skjematype + "-" + valgType + "-" + valgÃ¥r + "-" + tidspunkt + ".json"))
{
 37|        fw.write(skjemaInnhold.getJsonObject().toString());
```

5.23 LoadSave.java

Issue A7D0F193DAA6670B4D83A6A8EF43DEA9:

Sink :
File valg_database/src/main/java/no/valg/valgnatt/database/forberedvalg/LoadSave.java 67

```
 64|Files.createDirectories(new File(dirName).toPath());
 65|String fullName = dirName + csv.getFileName();
 66|LOG.info("Skriver " + fullName);
>67|try (FileWriter fileWriter = new FileWriter(fullName)) {
 68|    ICsvBeanWriter beanWriter = new CsvBeanWriter(new BufferedWriter(fileWriter), csv.getPreference());  69|
String[] header = csv.getHeader();
 70|    CellProcessor[] processors = csv.getProcessors();
```

5.24 V20150000_002__Baseline_Load.java

Issue EF3D81921D5338B108640A02A66BF86E:

Sink :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 39

```
 36|Resource[] resources = getResources();
 37|
 38|LOG.info("Laster inn data i tilstand");
>39|try (Statement statement = connection.createStatement()) {
 40|    // hvilke grunnlagsdata som blir brukt skrives her og sjekkes i GrunnlagsdataCache
 41|    statement.executeUpdate("insert into tilstand values('initialdata', '" + initialdata + "')");
```

[Medium] Privacy Violation / Autocomplete

Security Features

Analysis: Suspicious

Abstract

Autocompletion of forms allows some browsers to retain sensitive information in their history.

Explanation

With autocompletion enabled, some browsers retain user input across sessions, which could allow someone using the computer after the initial user to see information previously submitted.

Recommendation

Explicitly disable autocompletion on forms or sensitive inputs. By disabling autocompletion, information previously entered will not be presented back to the user as they type. It will also disable the "remember my password" functionality of most major browsers.

Example 1: In an HTML form, disable autocompletion for all input fields by explicitly setting the value of the `autocomplete` attribute to `off` on the `form` tag.

```
<form method="post" autocomplete="off">
      Address: <input name="address" />
      Password: <input name="password" type="password" />
</form>
```

Example 2: Alternatively, disable autocompletion for specific input fields by explicitly setting the value of the `autocomplete` attribute to `off` on the corresponding tags.

```
<form method="post">
      Address: <input name="address" />
      Password: <input name="password" type="password" autocomplete="off"/>   </form>
```

Note that the default value of the `autocomplete` attributed is `on`. Therefore do not omit the attribute when dealing with sensitive inputs.

5.25 login.jsp

Issue 5A2A6C4E9DE3F54518F5CB5FBBFFAFD0:

Sink :
File valg_overvaking/src/main/webapp/login.jsp 28

```
 25|</tr>
 26|<tr>
 27|    <td>Passord:</td>
>28|    <td><label><input type="password" name="j_password"></label></td>  29|</tr>
 30|<tr>
 31|    <td><br><input type="submit" value="Logg inn"></td>
```

[Medium] Redundant Null Check

Code Quality

Analysis: Reliability Issue

Abstract

The program can dereference a null pointer, thereby causing a null pointer exception.

Explanation

Null pointer exceptions usually occur when one or more of the programmer's assumptions is violated. Specifically, dereference-after-check errors occur when a program makes an explicit check for null, but proceeds to dereference the object when it is known to be null. Errors of this type are often the result of a typo or programmer oversight.

Most null pointer issues result in general software reliability problems, but if attackers can intentionally cause the program to dereference a null pointer, they can use the resulting exception to mount a denial of service attack or to cause the application to reveal debugging information that will be valuable in planning subsequent attacks.

Example 1: In the following code, the programmer confirms that the variable `foo` is `null` and subsequently dereferences it erroneously. If `foo` is `null` when it is checked in the `if` statement, then a null dereference will occur, thereby causing a null pointer exception.

```
if (foo == null) {
foo.setBar(val); ...
}
```

Recommendation

Implement careful checks before dereferencing objects that might be `null`. When possible, abstract null checks into wrappers around code that manipulates resources to ensure that they are applied in all cases and to minimize the places where mistakes can occur.

5.26 MandatBeregningService.java

Issue 974956CCA81CF12AF8A5D0368C726FFC:

Source :
File valg_backend/src/main/java/no/valg/valgnatt/backend/mandatberegning/MandatBeregningService.java 59 **Action**
Compared with null : resultater

```
 56|for (Mandatberegning rad : mandatberegningerLandet) {
 57|    String partikode = rad.getId().getPartikode();
 58|    Map<String, Integer> utjMandaterPrParti = null;
>59|    if (resultater != null && resultater.getPartiPrFylke() != null) {
 60|        utjMandaterPrParti = resultater.getPartiPrFylke().values().stream().collect(Collectors.groupingBy(parti ->
parti, Collectors.summingInt(parti -> 1)));
 61|        rad.setStemmervinneUtj(resultater.getVinnePrParti().get(partikode));  62|
rad.setStemmermisteUtj(resultater.getTapePrParti().get(partikode));
```

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/mandatberegning/MandatBeregningService.java 72 **Action**
Dereferenced : resultater

```
 69|for (Mandatberegning rad : mandatberegningerFylker) {
 70|    String fylkeNr = rad.getId().getValgdistriktnr();
 71|    String partikode = rad.getId().getPartikode();
>72|    rad.setAntallUtj(resultater.getPartiPrFylke().get(fylkeNr).equals(partikode) ? 1 : 0);
 73|    rad.setMandatrangUtj(resultater.getMandatnrPrPartiPrFylke().get(fylkeNr).getOrDefault(partikode, 0));  74|
rad.setRestkvotientUtj(BigDecimal.ZERO);
 75|    if (resultater.getKvotienterPrPartiPrFylke().get(fylkeNr).containsKey(partikode)) {
```

5.27 ValgTypeAar.java

Issue BD32F457DA3CB67A6EAE026FE89D7059:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/domene/ValgTypeAar.java 34
Action Compared with null : o

```
 32|@Override
 33|public boolean equals(Object o) {
>34|    return o != null && o.getClass() == ValgTypeAar.class || o.toString().equals(toString());
 35|}
 36|
 37|@Override
```

Sink :
File valg_database/src/main/java/no/valg/valgnatt/database/domene/ValgTypeAar.java 34
Action Dereferenced : o

```
 32|@Override
 33|public boolean equals(Object o) {
>34|    return o != null && o.getClass() == ValgTypeAar.class || o.toString().equals(toString());
 35|}
 36|
 37|@Override
```

[Medium] Resource Injection

Input Validation and Representation

Analysis: Not an Issue / Suspicious

Abstract

Allowing user input to control resource identifiers could enable an attacker to access or modify otherwise protected system resources.

Explanation

A resource injection issue occurs when the following two conditions are met:

1. An attacker is able to specify the identifier used to access a system resource.

For example, an attacker may be able to specify a port number to be used to connect to a network resource.

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 transmit sensitive information to a thirdparty server.

Note: Resource injections involving resources stored on the file system are reported in a separate category named path manipulation. See the path manipulation description for further details of this vulnerability.

Example 1: The following code uses a port number read from an HTTP request to create a socket.

```
String remotePort = request.getParameter("remotePort");
...
ServerSocket srvr = new
ServerSocket(remotePort); Socket skt =
srvr.accept(); ...
```

Some think that in the mobile world, classic web application vulnerabilities, such as resource injection, 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 2: The following code uses a URL read from an Android intent to load the page in `WebView`.

```
...
WebView webview = new WebView(this);
setContentView(webview);
        String url = this.getIntent().getExtras().getString("url"); webview.loadUrl(url); ...
```

The kind of resource affected by user input indicates the kind of content that may be dangerous. For example, data containing special characters like period, slash, and backslash are risky when used in methods that interact with the file system. Similarly, data that contains URLs and URIs is risky for functions that create remote connections.

Recommendation

The best way to prevent resource injection 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.

5.28 visRapport.jsp

Issue 04D0758CC71239437E44F5D54B4247AF:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 14
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
 11|String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";
 13|String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
>14|String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
 15|StringBuffer xmlFormatert = new StringBuffer("");
 16|if (rapportnr != null && rapportnr.length() > 0) {
 17|    String rapporturl = "/" + rapportnr;
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 38
Action InCall URL(0)
Rule id 2DEE27D8-C41F-48FC-8B40-FA60B403AEAE
Fact Call Direct : java.net.URL.URL
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
 37|try {
>38|    URL rapport = new URL(rapporturl);
 39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
 40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

Issue 087505BDCD5B01399AA3E8408B369725:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/Configuration.java 97
Action OutCall getRequiredProperty(return)
Rule id EBEF7D93-D788-4528-A22A-8CA0988BB4A9
Fact Call Direct : org.springframework.core.env.PropertyResolver.getRequiredProperty

```
  94|    if (env.containsProperty(key)) {
  95|        return env.getRequiredProperty(key, targetType);
  96|    }
> 97|    return env.getRequiredProperty("global/" + key, targetType);
  98|}
99|
 100|private <T> T getOptionalProperty(String key, Class<T> targetType, T defaultValue) {
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 38
Action InCall URL(0)
Rule id 2DEE27D8-C41F-48FC-8B40-FA60B403AEAE
Fact Call Direct : java.net.URL.URL
Fact TaintFlags ARGS, ENVIRONMENT, PROPERTY

```
 36|rapporturl = receiverurl + rapporturl;
37|try {
>38|    URL rapport = new URL(rapporturl);
 39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

Issue 16FBC75FFC06876818DE69BE73BB8B9F:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 11
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1
Fact Call Direct : javax.servlet.ServletRequest.getParameter

```
  9|<%
 10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
>11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|
String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|    String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 38
Action InCall URL(0)
Rule id 2DEE27D8-C41F-48FC-8B40-FA60B403AEAE
Fact Call Direct : java.net.URL.URL
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
 37|try {
>38|    URL rapport = new URL(rapporturl);
 39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

Issue 55AD92A9850478A089E876972BF811C9:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 12
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1
Fact Call Direct : javax.servlet.ServletRequest.getParameter

```
  9|<%
 10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
>12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|
String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|    String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";  15|
StringBuffer xmlFormatert = new StringBuffer("");
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 38
Action InCall URL(0)
Rule id 2DEE27D8-C41F-48FC-8B40-FA60B403AEAE
Fact Call Direct : java.net.URL.URL
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
37|try {
>38|    URL rapport = new URL(rapporturl);
 39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
 40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

Issue BFD6B8EF328638D72B49CCEA000BDA78:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/Configuration.java 95
Action OutCall getRequiredProperty(return)
Rule id EBEF7D93-D788-4528-A22A-8CA0988BB4A9
Fact Call Direct : org.springframework.core.env.PropertyResolver.getRequiredProperty

```
 93|private <T> T getRequiredProperty(String key, Class<T> targetType) {
 94|    if (env.containsProperty(key)) {
>95|        return env.getRequiredProperty(key, targetType);
 96|    }
 97|    return env.getRequiredProperty("global/" + key, targetType);
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 38
Action InCall URL(0)
Rule id 2DEE27D8-C41F-48FC-8B40-FA60B403AEAE
Fact Call Direct : java.net.URL.URL
Fact TaintFlags ARGS, ENVIRONMENT, PROPERTY

```
 36|rapporturl = receiverurl + rapporturl;
 37|try {
>38|    URL rapport = new URL(rapporturl);
 39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
 40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

Issue E2B19CF1AB562E6BA7C5880EA3CF3AC9:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 13
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1
Fact Call Direct : javax.servlet.ServletRequest.getParameter

```
 10|String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : ""; >13|String
kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
 15|StringBuffer xmlFormatert = new StringBuffer("");
 16|if (rapportnr != null && rapportnr.length() > 0) {
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 38
Action InCall URL(0)
Rule id 2DEE27D8-C41F-48FC-8B40-FA60B403AEAE
Fact Call Direct : java.net.URL.URL
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
 37|try {
>38|    URL rapport = new URL(rapporturl);
 39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
 40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

Issue F583B3415EB65E79E352383641C8A999:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 10
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1

```
  7|<%@ include file="/jspf/top.jspf" %>
  8|
  9|<%
>10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|
String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 38
Action InCall URL(0)
Rule id 2DEE27D8-C41F-48FC-8B40-FA60B403AEAE
Fact Call Direct : java.net.URL.URL
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
 37|try {
>38|    URL rapport = new URL(rapporturl);
 39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
 40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
```

[Medium] Server-Side Request Forgery

Input Validation and Representation

Analysis: Suspicious

Abstract

The application initiates a network connection to a third-party system using user-controlled data to craft the resource URI.

Explanation

A Server-Side Request Forgery occurs when an attacker may influence a network connection made by the application server. The network connection will originate from the application server's internal IP and an attacker will be able to use this connection to bypass network controls and scan or attack internal resources that are not otherwise exposed.

Even though the data in this case is a number, it is unvalidated and thus still considered malicious, hence the vulnerability is still reported but with reduced priority values.

Example: In the following example, an attacker will be able to control the URL the server is connecting to.

```
String url = request.getParameter("url");
CloseableHttpClient httpclient = HttpClients.createDefault();
HttpGet httpGet = new HttpGet(url);
CloseableHttpResponse response1 = httpclient.execute(httpGet);
```

The attacker's ability to hijack the network connection will depend upon the specific part of the URI that can be controlled, and upon libraries used to establish the connection. For example, controlling the URI scheme will let the attacker use protocols different from `http` or `https` like:

- up:// - ldap:// - jar:// - gopher:// - mailto:// - ssh2:// - telnet:// - expect://

An attacker will be able to leverage this hijacked network connection to perform the following attacks:

- Port Scanning of intranet resources. - Bypass firewalls. - Attack vulnerable programs running on the application server or on the intranet. - Attack internal/external web applications using Injection attacks or CSRF. - Access local files using file:// scheme. - On Windows systems, file:// scheme and UNC paths can allow an attacker to scan and access internal shares. - Perform a DNS cache poisoning attack.

Recommendation

Do not establish network connections based on user-controlled data and ensure that the request is being sent to the expected destination. If user data is necessary to build the destination URI, use 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.

Also, if required, make sure that the user input is only used to specify a resource on the target system but that the URI scheme, host, and port is controlled by the application. This way the damage that an attacker is able to do will be significantly reduced.

5.29 visRapport.jsp

Issue 29C816783899D342FD604256ED7D91C1:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 10
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
  7|<%@ include file="/jspf/top.jspf" %>
  8|
  9|<%
>10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|
String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 39
Action InCall openConnection(this)
Rule id 33DDCCB9-248C-46E8-BF4E-0A8DFA006946
Fact Call Direct : java.net.URL.openConnection
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
37|try {
 38|    URL rapport = new URL(rapporturl);
>39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
42|    String line;
```

Issue 93500F8DF0340DEEE0BF6EEDD9776495:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 11
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
  9|<%
 10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
>11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";  13|
String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|    String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 39
Action InCall openConnection(this)
Rule id 33DDCCB9-248C-46E8-BF4E-0A8DFA006946
Fact Call Direct : java.net.URL.openConnection
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
 37|try {
 38|    URL rapport = new URL(rapporturl);
>39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
42|    String line;
```

Issue 9E31E9818579F8492F5514490ED7C8F1:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 12
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct :
javax.servlet.ServletRequest.getParameter

```
  9|<%
 10|    String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|    String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
>12|    String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";
 13|    String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|    String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";  15|
StringBuffer xmlFormatert = new StringBuffer("");
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 39
Action InCall openConnection(this)
Rule id 33DDCCB9-248C-46E8-BF4E-0A8DFA006946
Fact Call Direct : java.net.URL.openConnection

Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
37|try {
 38|    URL rapport = new URL(rapporturl);
>39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
42|    String line;
```

Issue B575C3169893CCC6F3C81A8D7166481F:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 13
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
 10|String rapportnr = request.getParameter("rapportnr") != null ? request.getParameter("rapportnr") : "";
 11|String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";
>13|String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
 15|StringBuffer xmlFormatert = new StringBuffer("");
 16|if (rapportnr != null && rapportnr.length() > 0) {
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 39
Action InCall openConnection(this)
Rule id 33DDCCB9-248C-46E8-BF4E-0A8DFA006946
Fact Call Direct : java.net.URL.openConnection
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
37|try {
 38|    URL rapport = new URL(rapporturl);
>39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
42|    String line;
```

Issue F4E58E0AF605E9EF937C92B0477FA1E1:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 14
Action OutCall getParameter(return)
Rule id D312DFA3-EF02-46A5-A25B-29D218E96EF1 **Fact** Call Direct : javax.servlet.ServletRequest.getParameter

```
 11|String fylkenr = request.getParameter("fylkenr") != null ? request.getParameter("fylkenr") : "";
 12|String kommnr = request.getParameter("kommunenr") != null ? request.getParameter("kommunenr") : "";
 13|String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
>14|String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
15|StringBuffer xmlFormatert = new StringBuffer("");
 16|if (rapportnr != null && rapportnr.length() > 0) {
 17|    String rapporturl = "/" + rapportnr;
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 39
Action InCall openConnection(this)
Rule id 33DDCCB9-248C-46E8-BF4E-0A8DFA006946
Fact Call Direct : java.net.URL.openConnection
Fact TaintFlags WEB, XSS

```
 36|rapporturl = receiverurl + rapporturl;
37|try {
 38|    URL rapport = new URL(rapporturl);
>39|    HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
40|    conn.connect();
 41|    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
42|    String line;
```

[Medium] Session Fixation

Time and State

Analysis: Suspicious

Abstract

Authenticating a user without invalidating any existing session identifier gives an attacker the opportunity to steal authenticated sessions.

Explanation

Session fixation vulnerabilities occur when:

1. A web application authenticates a user without first invalidating the existing session, thereby continuing to use the session already associated with the user.

2. An attacker is able to force a known session identifier on a user so that, once the user authenticates, the attacker has access to the authenticated session.

In the generic exploit of session fixation vulnerabilities, an attacker creates a new session on a web application and records the associated session identifier. The attacker then causes the victim to authenticate against the server using that session identifier, giving the attacker access to the user's account through the active session.

Example: The following example shows a snippet of code from a J2EE web application where the application authenticates users with a direct post to the `j_security_check`, which typically does not invalidate the existing session before processing the login request.

```
<form method="POST" action="j_security_check">
<input type="text" name="j_username">
<input type="text" name="j_password"> </form>
```

In order to exploit the code above, an attacker could first create a session (perhaps by logging into the application) from a public terminal, record the session identifier assigned by the application, and reset the browser to the login page. Next, a victim sits down at the same public terminal, notices the browser open to the login page of the site, and enters credentials to authenticate against the application. The code responsible for authenticating the victim continues to use the pre-existing session identifier, now the attacker simply uses the session identifier recorded earlier to access the victim's active session, providing nearly unrestricted access to the victim's account for the lifetime of the session.

Even given a vulnerable application, the success of the specific attack described here is dependent on several factors working in the favor of the attacker: access to an unmonitored public terminal, the ability to keep the compromised session active and a victim interested in logging into the vulnerable application on the public terminal. In most circumstances, the first two challenges are surmountable given a sufficient investment of time. Finding a victim who is both using a public terminal and interested in logging into the vulnerable application is possible as well, so long as the site is reasonably popular. The less well known the site is, the lower the odds of an interested victim using the public terminal and the lower the chance of success for the attack vector described above.

The biggest challenge an attacker faces in exploiting session fixation vulnerabilities is inducing victims to authenticate against the vulnerable application using a session identifier known to the attacker. In the example above, the attacker did this through a direct method that is not subtle and does not scale suitably for attacks involving less well-known web sites. However, do not be lulled into complacency; attackers have many tools in their belts that help bypass the limitations of this attack vector. The most common technique employed by attackers involves taking advantage of cross-site scripting or HTTP response splitting vulnerabilities in the target site [1]. By tricking the victim into submitting a malicious request to a vulnerable application that reflects JavaScript or other code back to the victim's browser, an attacker may create a cookie that will cause the victim to reuse a session identifier controlled by the attacker.

It is worth noting that cookies are often tied to the top level domain associated with a given URL. If multiple applications reside on the same top level domain, such as bank.example.com and recipes.example.com, a vulnerability in one application can allow an attacker to set a cookie with a fixed session identifier that will be used in all interactions with any application on the domain example.com [2].

Other attack vectors include DNS poisoning and related network based attacks where an attacker causes the user to visit a malicious site by redirecting a request for a valid site. Network based attacks typically involve a physical presence on the victim's network or control of a compromised machine on the network, which makes them harder to exploit remotely, their significance should not be overlooked. Less secure session management mechanisms, such as the default implementation in Apache Tomcat, allow session identifiers normally expected in a cookie to be specified on the URL as well, which enables an attacker to cause a victim to use a fixed session identifier simply by emailing a malicious URL.

Recommendation

In order to prevent session fixation, a web-based application must issue a new session identifier at the same time it authenticates a user. Many application servers make this more difficult by providing separate facilities for managing authorization and session management. For example, an application that posts authentication data directly to the URL `j_security_check` under the Tomcat Servlet container will cause Tomcat to perform authentication without issuing a new session identifier.

The only effective solution is to implement proprietary code to perform authentication and ensure that you invalidate an existing session before processing login requests. Any existing user information stored in the session can be migrated to the new session safely, so long as the associated session identifier changes. Keep in mind that the order of operations is important: the existing session must be invalidated before the login request is processed. If the session is invalidated after the user is logged in, a race condition exists where the attacker knows the session identifier for some period after authentication.

5.30 login.jsp

Issue 74E2571ED84030A75DA8EC9C69F09E65:

Sink :
File valg_overvaking/src/main/webapp/login.jsp 20-34
Action j_security_check

```
 17|                    <td><strong>Skriv inn brukernavn og passord</strong></td>
 18|                </tr>
 19|            </table>
>20|            <form method="POST" action="j_security_check">
>21|                <table>
>22|                    <tr>
>23|                        <td>Brukernavn:</td>
>24|                        <td><label><input type="text" name="j_username"></label></td>
>25|                    </tr>
>26|                    <tr>
>27|                        <td>Passord:</td>
>28|                        <td><label><input type="password" name="j_password"></label></td>
>29|                    </tr>
>30|                    <tr>
>31|                        <td><br><input type="submit" value="Logg inn"></td>
>32|                    </tr>
>33|                </table>
>34|            </form>
 35|        </td>
 36|    </tr>
 37|</table>
```

[Medium] System Information Leak / Incomplete Servlet Error Handling

Encapsulation

Analysis: Suspicious

Abstract

If a Servlet fails to catch all exceptions, it might reveal debugging information that will help an adversary form a plan of attack.

Explanation

When a Servlet throws an exception, the default error response the Servlet container sends back to the user typically includes debugging information. This information is of great value to an attacker. For example, a stack trace might show the attacker a malformed SQL query string, the type of database being used, and the version of the application container. This information enables the attacker to target known vulnerabilities in these components.

Example 1: In the following method a DNS lookup failure will cause the Servlet to throw an exception.

```
protected void doPost (HttpServletRequest req,
HttpServletResponse res)
              throws IOException {
String ip = req.getRemoteAddr();
    InetAddress addr = InetAddress.getByName(ip);
...
    out.println("hello " + addr.getHostName());
}
```

Example 2: The following method will throw a `NullPointerException` if the parameter "name" is not part of the request.

```
protected void doPost (HttpServletRequest
req,                     HttpServletResponse
res)               throws IOException {
String name = getParameter("name");     ...
    out.println("hello " + name.trim()); }
```

Recommendation

All top-level Servlet methods should catch `Throwable`, thereby minimizing the chance that the Servlet's error response mechanism is invoked.

Example 3: The method from Example 1 should be rewritten as follows:

```
protected void doPost (HttpServletRequest req,
HttpServletResponse res) {       try {
          String ip = req.getRemoteAddr();
          InetAddress addr = InetAddress.getByName(ip);
      ...
          out.println("hello " + addr.getHostName());
      }catch (Throwable t) {
          logger.error("caught throwable at top level", t);
      }
  }
}
```

5.31 MottakFraBackendServlet.java

Issue ACCF508329E5A4419EEF0FB9D255F240:

Sink :
File valg_frontend/src/main/java/no/valg/valgnatt/frontend/MottakFraBackendServlet.java 40-63
Fact Name: no.valg.valgnatt.frontend.MottakFraBackendServlet.doPost

```
    @Override
    public void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        try {
            StringBuilder sb = new StringBuilder();
            BufferedReader br = req.getReader();
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line).append('\n');
            }
            br.close();

            String reply;
            if (sb.toString().startsWith(STATUS)) {
                reply = kontroller.prosesserStatus(sb.toString());
            } else {
                reply = kontroller.prosesserRapporterFraBackend(sb.toString());
            }

            res.setContentType("text/plain");
            res.getWriter().println(reply);
        } catch (Exception e) {
            LOG.error("Feil oppstod: " + e, e);
            throw new ServletException("Feil oppstod i MottakFraBackendServlet.doPost()", e);
```

[Medium] Unreleased Resource / Streams

Code Quality

Analysis: Reliability Issue

Abstract

The program can potentially fail to release a system resource.

Explanation

The program can potentially fail to release a system resource.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems, but if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool.

Example: The following method never closes the file handle it opens. The `finalize()` method for `FileInputStream` eventually calls `close()`, but there is no guarantee as to how long it will take before the `finalize()` method will be invoked. In a busy environment, this can result in the JVM using up all of its file handles.

```
private void processFile(String fName) throws FileNotFoundException, IOException {
  FileInputStream fis = new FileInputStream(fName);
int sz;
  byte[] byteArray = new byte[BLOCK_SIZE];
while ((sz = fis.read(byteArray)) != -1) {
processBytes(byteArray, sz);
  }
}
```

Recommendation

1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short

period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases.

Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network to a database, for example), then the thread that is executing the `finalize()` method will hang.

2. Release resources in a `finally` block. The code for the Example should be rewritten as follows:

```
public void processFile(String fName) throws FileNotFoundException, IOException {
  FileInputStream fis;
try {
    fis = new FileInputStream(fName);
int sz;
    byte[] byteArray = new byte[BLOCK_SIZE];
while ((sz = fis.read(byteArray)) != -1) {
processBytes(byteArray, sz);
    }   }
finally {
    if (fis != null) {
safeClose(fis);
    }
  }
}


public static void safeClose(FileInputStream fis)
{   if (fis != null) {     try {
      fis.close();
    } catch (IOException e) {
log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the stream. Presumably this helper function will be reused whenever a stream needs to be closed.

Also, the `processFile` method does not initialize the `fis` object to null. Instead, it checks to ensure that `fis` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `fis` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `fis` is initialized to `null` in a more complex method, cases in which `fis` is used without being initialized will not be detected by the compiler.

5.32 ConfigurationCopy.java

Issue A52ADF3AB0354FC76AB3B8E18FEA1009:

Source :
File valg_utils/src/main/java/no/valg/valgnatt/utils/ConfigurationCopy.java 15 **Action** BranchNotTaken Branch not taken: ((args.length) == 1)

```
 12|public class ConfigurationCopy {
 13|
 14|    public static void main(String[] args) throws Exception {
>15|        if (args.length != 1) {
 16|            throw new RuntimeException("Required command-line argument missing");
```

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/ConfigurationCopy.java 34
Action InCall fw = new FileWriter(...)
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 31|    File file = new File(args[0]);
 32|    //noinspection ResultOfMethodCallIgnored
 33|    file.getParentFile().mkdirs();
>34|    FileWriter fw = new FileWriter(file);
 35|    fw.write(sb.toString());
 36|    fw.close();
```

5.33 ConfigurationLoader.java

Issue 21605BB02E81B11BBE3F3B296FC08658:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/util/ConfigurationLoader.java 20
Action BranchNotTaken Branch not taken: (configuration != null)

```
 18|public static void load() {
 19|    String configuration = System.getProperty("configuration");
>20|    if (configuration == null) {
 21|        throw new RuntimeException("Det mangler en VM-option ved kjÃ¸ring av dette programmet som angir en
konfigurasjonsfil. Syntaks: -Dconfiguration=classpath:<filnavn>");
 22|    }
 23|    Resource resource = new DefaultResourceLoader().getResource(configuration);
```

Sink :
File valg_database/src/main/java/no/valg/valgnatt/database/util/ConfigurationLoader.java 25
Action Assign is = getInputStream()
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 23|Resource resource = new DefaultResourceLoader().getResource(configuration);  24|try {
>25|    InputStream is = resource.getInputStream();
 26|    properties.load(is);
 27|    is.close();
 28|} catch (IOException e) {
```

5.34 LoadSave.java

Issue 0506D22EE764C5ED74DDE7F9E07AD9F2:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/forberedvalg/LoadSave.java 42
Action BranchNotTaken Branch not taken: (resourceAsStream == null)

```
 40|Reader reader;
 41|InputStream resourceAsStream = LoadSave.class.getClassLoader().getResourceAsStream(fullName); >42|if
(resourceAsStream != null) {  43|    reader = new InputStreamReader(resourceAsStream);  44|} else {
 45|    reader = new BufferedReader(new FileReader(fullName));
```

Sink :
File valg_database/src/main/java/no/valg/valgnatt/database/forberedvalg/LoadSave.java 45
Action InCall reader = new BufferedReader(new java.io.FileReader())
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 42|if (resourceAsStream != null) {  43|    reader = new
InputStreamReader(resourceAsStream);  44|} else {
>45|    reader = new BufferedReader(new FileReader(fullName));
 46|}

 47|
 48|ICsvBeanReader beanReader = new CsvBeanReader(reader, csv.getPreference());
```

Issue 8E0EE4AB57484BD4CCFFB3050F1C5E8E:

Source :
File valg_database/src/main/java/no/valg/valgnatt/database/forberedvalg/LoadSave.java 41
Action Assign resourceAsStream = getResourceAsStream(...) **Rule id** 74714BFC-EDF7-445B-8672-0996214D5845

```
 38|LOG.info("Leser " + fullName);
 39|
 40|Reader reader;
>41|InputStream resourceAsStream = LoadSave.class.getClassLoader().getResourceAsStream(fullName);
 42|if (resourceAsStream != null) {  43|    reader = new InputStreamReader(resourceAsStream);  44|} else {
```

Sink :
File valg_database/src/main/java/no/valg/valgnatt/database/forberedvalg/LoadSave.java 43
Action InCall reader = new InputStreamReader(resourceAsStream)
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 40|Reader reader;
 41|InputStream resourceAsStream = LoadSave.class.getClassLoader().getResourceAsStream(fullName);
 42|if (resourceAsStream != null) { >43|    reader = new InputStreamReader(resourceAsStream);
 44|} else {
 45|    reader = new BufferedReader(new FileReader(fullName));
```

5.35 MainMottak.java

Issue 83DA253345CD22F079613857F3E9BC0E:

Source :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 93
Action Assign is = getErrorStream()
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 90|conn.connect();
 91|responseCode = conn.getResponseCode();
 92|responseMessage = conn.getResponseMessage();
>93|InputStream is = conn.getErrorStream();
 94|if (is == null) {
 95|    is = conn.getInputStream();
```

Sink :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 97
Action InCall br = new BufferedReader(new java.io.InputStreamReader())
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
  94|if (is == null) {
  95|    is = conn.getInputStream();
  96|}
> 97|BufferedReader br = new BufferedReader(new InputStreamReader(is));
  98|String line;
  99|StringBuilder sb = new StringBuilder();
 100|while ((line = br.readLine()) != null) {
```

5.36 StartupServletContextListener.java

Issue AD428DF71D115DEDADA6C1238B863623:

Source :
File valg_common/src/main/java/no/valg/valgnatt/common/StartupServletContextListener.java 36
Action Assign is = getResourceAsStream(...)
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 35|Properties properties = new Properties();
>36|InputStream is = StartupServletContextListener.class.getResourceAsStream("/version.properties");
 37|if (is != null) {
 38|    try {
 39|        properties.load(is);
```

Sink :
File valg_common/src/main/java/no/valg/valgnatt/common/StartupServletContextListener.java 36
Action Assign is = getResourceAsStream(...)
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 35|Properties properties = new Properties();
>36|InputStream is = StartupServletContextListener.class.getResourceAsStream("/version.properties");
 37|if (is != null) {
 38|    try {
 39|        properties.load(is);
```

5.37 StemmeInnsender.java

Issue 14EADA1C0DC6647A14493859616D2F8F:

Source :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 94
Action BranchTaken Branch taken: (i < resourcesFiltered.size())

```
 91|boolean vent = false;
 92|long waitedMs = 0;
 93|String file = null;
>94|for (int i = 0; i < resourcesFiltered.size(); i++) {
 95|    Resource resource = resourcesFiltered.get(i);
 96|    file = resource.getFilename();
 97|    String ventetid = null;
```

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 102 **Action** InOutCall ?.send(?, resource.getInputStream())
Trace id 0

```
  99|        ventetid = vent(hastighet, forrigeFilTms, waitedMs, file);
 100|    }
  101|    loggLinjePrRunde(file, i, ventetid);
 >102|    waitedMs = send(host, resource.getInputStream());
  103|    forrigeFilTms = tms(filnavnTidspunkt(file));
 104|    vent = true;
```

5.38 Stemmeskjema2Filsystem.java

Issue 979F50B6EA2C240AF7480F4F15F9081B:

Source :
File valg_utils/src/main/java/no/valg/valgnatt/utils/Stemmeskjema2Filsystem.java 63
Action BranchTaken Branch taken: rs.next()

```
 60|int counter = 0;
 61|String sql = "SELECT opprettet_tid, valgtype, kommnr, kretsnr, data FROM stemmeskjema";
 62|try (Statement s = c.createStatement(); ResultSet rs = s.executeQuery(sql)) {
>63|    while (rs.next()) {
 64|        Date opprettet = rs.getTimestamp("opprettet_tid");
 65|        String valgtype = rs.getString("valgtype");
 66|        String kommnr = rs.getString("kommnr");
```

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/Stemmeskjema2Filsystem.java 77
Action InCall out = new BufferedWriter(new java.io.OutputStreamWriter()) **Rule id** 74714BFC-EDF7-445B-8672-0996214D5845

```
 74|File file = new File(f.getAbsoluteFile() + File.separator + sdf3.format(opprettet) + "-" + kretsnr + ".json");
75|if (!file.exists()) {
 76|    LOG.log(Level.INFO, "File " + file.getAbsoluteFile() + " finnes ikke, mÃ¥ lages");
>77|    BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file), "UTF-8"));
 78|    out.write(toPrettyFormat(skjema));
 79|    out.flush();
 80|    out.close();
```

Issue 9F941BB0715D9983FC91EBD76797529B:

Source :
File valg_utils/src/main/java/no/valg/valgnatt/utils/Stemmeskjema2Filsystem.java 63
Action BranchTaken Branch taken: rs.next()

```
 60|int counter = 0;
 61|String sql = "SELECT opprettet_tid, valgtype, kommnr, kretsnr, data FROM stemmeskjema";
 62|try (Statement s = c.createStatement(); ResultSet rs = s.executeQuery(sql)) {
>63|    while (rs.next()) {
 64|        Date opprettet = rs.getTimestamp("opprettet_tid");
 65|        String valgtype = rs.getString("valgtype");
 66|        String kommnr = rs.getString("kommnr");
```

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/Stemmeskjema2Filsystem.java 77
Action InCall new OutputStreamWriter(new java.io.FileOutputStream(), ...) **Rule id** 74714BFC-EDF7-445B-8672-0996214D5845

```
 74|File file = new File(f.getAbsoluteFile() + File.separator + sdf3.format(opprettet) + "-" + kretsnr + ".json");
 75|if (!file.exists()) {
 76|    LOG.log(Level.INFO, "File " + file.getAbsoluteFile() + " finnes ikke, mÃ¥ lages");
>77|    BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file), "UTF-8"));
 78|    out.write(toPrettyFormat(skjema));
 79|    out.flush();
 80|    out.close();
```

5.39 V20150000_002__Baseline_Load.java

Issue 751191482833BC87E58B91BAE443E0F1:

Source :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 49
Action BranchTaken Branch taken: (resource~iterator < resource~iterator~len)

```
 47|BaseConnection base = getBaseConnection(connection);
 48|CopyManager cm = new CopyManager(base);
>49|for (Resource resource : resources) {
 50|    BufferedReader br = new BufferedReader(new InputStreamReader(resource.getInputStream()));
 51|    String colNames = br.readLine().replace(';', ',');
52|    br.close();
```

Sink :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 53
Action Assign is = getInputStream()

Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 50|BufferedReader br = new BufferedReader(new InputStreamReader(resource.getInputStream()));
51|String colNames = br.readLine().replace(';', ',');
 52|br.close();
>53|InputStream is = resource.getInputStream();
 54|String tableName = resource.getFilename().substring(0, resource.getFilename().indexOf('.'));
 55|LOG.info("Laster inn data i " + tableName);
 56|cm.copyIn("COPY " + tableName + " (" + colNames + ") FROM STDIN WITH (FORMAT CSV, HEADER, DELIMITER ';')", is);
```

Issue D015A24CBBD4762B7C45D9034AD944C0:

Source :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 49
Action BranchTaken Branch taken: (resource~iterator < resource~iterator~len)

```
 47|BaseConnection base = getBaseConnection(connection);
 48|CopyManager cm = new CopyManager(base);
>49|for (Resource resource : resources) {
 50|    BufferedReader br = new BufferedReader(new InputStreamReader(resource.getInputStream()));
 51|    String colNames = br.readLine().replace(';', ',');
52|    br.close();
```

Sink :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 50
Action InCall br = new BufferedReader(new java.io.InputStreamReader())
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 47|BaseConnection base = getBaseConnection(connection);
 48|CopyManager cm = new CopyManager(base);
 49|for (Resource resource : resources) {
>50|    BufferedReader br = new BufferedReader(new InputStreamReader(resource.getInputStream()));
 51|    String colNames = br.readLine().replace(';', ',');
 52|    br.close();
 53|    InputStream is = resource.getInputStream();
```

5.40 dashboard.jsp

Issue 9734D5513968EFE3135AC6F1963474A3:

Source :
File valg_overvaking/src/main/webapp/dashboard.jsp 46
Action Assign is = getResourceAsStream(...)
Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 44|StemmeskjemaInfo ssi = overvakingService.hentStemmeskjemaInfo();
 45|Properties properties = new Properties();
>46|InputStream is = StartupServletContextListener.class.getResourceAsStream("/version.properties");
 47|if (is != null) {
 48|    try {
 49|        properties.load(is);
```

Sink :
File valg_overvaking/src/main/webapp/dashboard.jsp 46
Action Assign is = getResourceAsStream(...)

Rule id 74714BFC-EDF7-445B-8672-0996214D5845

```
 44|StemmeskjemaInfo ssi = overvakingService.hentStemmeskjemaInfo();
 45|Properties properties = new Properties();
>46|InputStream is = StartupServletContextListener.class.getResourceAsStream("/version.properties");
 47|if (is != null) {
 48|    try {
 49|        properties.load(is);
```

5.41 visRapport.jsp

Issue 298F9B345B01D8C32AFE0D3B317CFF9D:

Source :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 16
Action BranchTaken Branch taken: (rapportnr != null)

```
 13|String kretsnr = request.getParameter("kretsnr") != null ? request.getParameter("kretsnr") : "";
 14|String bydelsnr = request.getParameter("bydelsnr") != null ? request.getParameter("bydelsnr") : "";
 15|StringBuffer xmlFormatert = new StringBuffer("");
>16|if (rapportnr != null && rapportnr.length() > 0) {
17|    String rapporturl = "/" + rapportnr;
 18|     if (fylkenr != null && fylkenr.length() > 0) {
 19|         rapporturl = rapporturl + "/" + fylkenr;
```

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 41
Action InCall br = new BufferedReader(new java.io.InputStreamReader()) **Rule id** 74714BFC-EDF7-445B-8672-0996214D5845

```
 38|URL rapport = new URL(rapporturl);
 39|HttpURLConnection conn = (HttpURLConnection) rapport.openConnection();
 40|conn.connect();
>41|BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));  42|String line;
 43|while ((line = br.readLine()) != null) {
 44|    xmlFormatert.append(line).append('\n');
```

[Low] Cross-Site Request Forgery

Encapsulation

Analysis: Suspicious

Abstract

Form posts must contain a user-specific secret in order to prevent an attacker from making unauthorized requests.

Explanation

A cross-site request forgery (CSRF) vulnerability occurs when: 1. A Web application uses session cookies.

2. The application acts on an HTTP request without verifying that the request was made with the user's consent.

A nonce is a cryptographic random value that is sent with a message to prevent replay attacks. If the request does not contain a nonce that proves its provenance, the code that handles the request is vulnerable to a CSRF attack (unless it does not change the state of the application). This means a Web application that uses session cookies has to take special precautions in order to ensure that an attacker can't trick users into submitting bogus requests. Imagine a Web application that allows administrators to create new accounts by submitting this form:

```
<form method="POST" action="/new_user" >
  Name of new user: <input type="text" name="username">
  Password for new user: <input type="password" name="user_passwd">
    <input type="submit" name="action" value="Create User">
</form>
```

An attacker might set up a Web site with the following:

```
<form method="POST" action="http://www.example.com/new_user">
  <input type="hidden" name="username" value="hacker">
  <input type="hidden" name="user_passwd" value="hacked">
</form>
<script>
  document.usr_form.submit();
</script>
```

If an administrator for `example.com` visits the malicious page while she has an active session on the site, she will unwittingly create an account for the attacker. This is a CSRF attack. It is possible because the application does not have a way to determine the provenance of the request. Any request could be a legitimate action chosen by the user or a faked action set up by an attacker. The attacker does not get to see the Web page that the bogus request generates, so the attack technique is only useful for requests that alter the state of the application.

Applications that pass the session identifier in the URL rather than as a cookie do not have CSRF problems because there is no way for the attacker to access the session identifier and include it as part of the bogus request.

CSRF is entry number five on the 2007 OWASP Top 10 list.

Recommendation

Applications that use session cookies must include some piece of information in every form post that the back-end code can use to validate the provenance of the request. One way to do that is to include a random request identifier or nonce, like this:

```
  RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, "/new_user");
body = addToPost(body, new_username);   body = addToPost(body, new_passwd);
body = addToPost(body, request_id);   rb.sendRequest(body, new
NewAccountCallback(callback));
```

Then the back-end logic can validate the request identifier before processing the rest of the form data. When possible, the request identifier should be unique to each server request rather than shared across every request for a particular session. As with session identifiers, the harder it is for an attacker to guess the request identifier, the harder it is to conduct a successful CSRF attack. The token should not be easily guessed and it should be protected in the same way that session tokens are protected, such as using SSLv3.

Additional mitigation techniques include:

Framework protection: Most modern web application frameworks embed CSRF protection and they will automatically include and verify CSRF tokens. **Use a Challenge-Response control:** Forcing the customer to respond to a challenge sent by the server is a strong defense against CSRF. Some of the challenges that can be used for this purpose are: CAPTCHAs, password re-authentication and one-time tokens. **Check HTTP Referer/Origin headers:** An attacker won't be able to spoof these headers while performing a CSRF attack. This makes these headers a useful method to prevent CSRF attacks. **Double-submit Session Cookie:** Sending the session ID Cookie as a hidden form value in addition to the actual session ID Cookie is a good protection against CSRF attacks. The server will check both values and make sure they are identical before processing the rest of the form data. If an attacker submits a form in behalf of a user, he won't be able to modify the session ID cookie value as per the same-origin-policy. **Limit Session Lifetime:** When accessing protected resources using a CSRF attack, the attack will only be valid as long as the session ID sent as part of the attack is still valid on the server. Limiting the Session lifetime will reduce the probability of a successful attack.

The techniques described here can be defeated with XSS attacks. Effective CSRF mitigation includes XSS mitigation techniques.

5.42 diverse.jsp

Issue F76537099CE49F26CF89A5025847FD3F:

Sink :
File valg_overvaking/src/main/webapp/jsp/diverse.jsp 105-108

```
 103|<h2>Send inn stemmeskjema</h2>
 104|
>105|<form id="form1" name="form1" method="post" action="/valg_mottak/" accept-charset="UTF-8">
>106|    <textarea name="stemmeskjemafradiversesiden" id="textarea" cols="120" rows="40"></textarea>
>107|    <input type="submit" name="button" id="button" onclick="return confirm('Er 100% sikker - du sender inn
stemmeskjema!!')" value="Send" />
>108|</form>
 109|
 110|</body>
 111|</html>
```

5.43 login.jsp

Issue D81E02434885DD5916A2A6C340373C63:

Sink :
File valg_overvaking/src/main/webapp/login.jsp 20-34

```
 17|                    <td><strong>Skriv inn brukernavn og passord</strong></td>
 18|                </tr>
 19|            </table>
>20|            <form method="POST" action="j_security_check">
>21|                <table>
>22|                    <tr>
>23|                        <td>Brukernavn:</td>
>24|                        <td><label><input type="text" name="j_username"></label></td>
>25|                    </tr>
>26|                    <tr>
>27|                        <td>Passord:</td>
>28|                        <td><label><input type="password" name="j_password"></label></td>
>29|                    </tr>
>30|                    <tr>
>31|                        <td><br><input type="submit" value="Logg inn"></td>
>32|                    </tr>
>33|                </table>
>34|            </form>
 35|        </td>
 36|    </tr>
 37|</table>
```

[Low] Denial of Service

Input Validation and Representation

Analysis: Reliability Issue

Abstract

An attacker could cause the program to crash or otherwise become unavailable to legitimate users.

Explanation

Attackers may be able to deny service to legitimate users by flooding the application with requests, but flooding attacks can often be defused at the network layer. More problematic are bugs that allow an attacker to overload the application using a small number of requests. Such bugs allow the attacker to specify the quantity of system resources their requests will consume or the duration for which they will use them.

Example 1: The following code allows a user to specify the amount of time for which a thread will sleep. By specifying a large number, an attacker may tie up the thread indefinitely. With a small number of requests, the attacker may deplete the application's thread pool.

```
  int usrSleepTime = Integer.parseInt(usrInput);
Thread.sleep(usrSleepTime);
```

Example 2: The following code reads a String from a zip file. Because it uses the `readLine()` method, it will read an unbounded amount of input. An attacker may take advantage of this code to cause an `OutOfMemoryException` or to consume a large amount of memory so that the program spends more time performing garbage collection or runs out of memory during some subsequent operation.

```
  InputStream zipInput = zipFile.getInputStream(zipEntry);
  Reader zipReader = new InputStreamReader(zipInput);
  BufferedReader br = new BufferedReader(zipReader);   String line = br.readLine();
```

Recommendation

Validate user input to ensure that it will not cause inappropriate resource utilization.

Example 3: The following code allows a user to specify the amount of time for which a thread will sleep just as in Example 1, but only if the value is within reasonable bounds.

```
  int usrSleepTime =
Integer.parseInt(usrInput);   if (usrSleepTime
>= SLEEP_MIN &&       usrSleepTime <=
SLEEP_MAX) {
    Thread.sleep(usrSleepTime);
  } else {
    throw new Exception("Invalid sleep duration");
  }
}
```

Example 4: The following code reads a String from a zip file just as in Example 2, but the maximum string length it will read is `MAX_STR_LEN` characters.

```
  InputStream zipInput = zipFile.getInputStream(zipEntry);
  Reader zipReader = new InputStreamReader(zipInput);
  BufferedReader br = new BufferedReader(zipReader);
  StringBuffer sb = new StringBuffer();
int intC;
  while ((intC = br.read()) != -1)
{     char c = (char) intC;     if
(c == '\n') {       break;     }
    if (sb.length() >= MAX_STR_LEN) {
throw new Exception("input too long");
    }
    sb.append(c);
  }
  String line = sb.toString();
```

5.44 MainMottak.java

Issue 04294667ADA8A66232334EAF4FAFE801:

Sink :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 100 **Action** InCall readLine()
Rule id 24023E22-D6C7-4D5C-B049-38B7EFC8B408

```
  97|BufferedReader br = new BufferedReader(new InputStreamReader(is));
  98|String line;
  99|StringBuilder sb = new StringBuilder();
>100|while ((line = br.readLine()) != null) {
 101|    sb.append(line).append('\n');
 102|}
 103|br.close();
```

5.45 MottakFraBackendServlet.java

Issue F3634F923EB497B0DC4468FAA5F97AE9:

Sink :
File valg_frontend/src/main/java/no/valg/valgnatt/frontend/MottakFraBackendServlet.java 45
Action InCall readLine()
Rule id 24023E22-D6C7-4D5C-B049-38B7EFC8B408

```
 42|StringBuilder sb = new StringBuilder();
 43|BufferedReader br = req.getReader();
 44|String line;
>45|while ((line = br.readLine()) != null) {
 46|    sb.append(line).append('\n');
 47|}
 48|br.close();
```

5.46 StemmeInnsender.java

Issue 721D35C52566E2340C2AD1F8263BA9E4:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 168
Action InCall readLine()

Rule id 24023E22-D6C7-4D5C-B049-38B7EFC8B408

```
 165|BufferedReader br = new BufferedReader(new InputStreamReader(is));
 166|StringBuilder content = new StringBuilder();
 167|String line;
>168|while ((line = br.readLine()) != null) {
 169|    content.append(line);
 170|}
 171|br.close();
```

5.47 V20150000_002__Baseline_Load.java

Issue B48D9A719EE2689B2BD1DFC116DBCFE4:

Sink :
File valg_database/src/main/java/db/migration/V20150000_002__Baseline_Load.java 51
Action InCall readLine()
Rule id 24023E22-D6C7-4D5C-B049-38B7EFC8B408

```
 48|CopyManager cm = new CopyManager(base);
 49|for (Resource resource : resources) {
 50|    BufferedReader br = new BufferedReader(new InputStreamReader(resource.getInputStream()));
>51|    String colNames = br.readLine().replace(';', ',');
 52|    br.close();
 53|    InputStream is = resource.getInputStream();
 54|    String tableName = resource.getFilename().substring(0, resource.getFilename().indexOf('.'));
```

5.48 visRapport.jsp

Issue 5E6C2B3CB9278C2D613AAFAAEB62B857:

Sink :
File valg_overvaking/src/main/webapp/jsp/visRapport.jsp 43
Action InCall readLine()
Rule id 24023E22-D6C7-4D5C-B049-38B7EFC8B408

```
 40|conn.connect();
 41|BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
42|String line;
>43|while ((line = br.readLine()) != null) {
 44|    xmlFormatert.append(line).append('\n');
45|}
 46|br.close();
```

[Low] Hardcoded Domain in HTML

Encapsulation

Analysis: Reliability Issue

Abstract

Including a script from another domain means that the security of this web page is dependent on the security of the other domain.

Explanation

Including executable content from another web site is a risky proposition. It ties the security of your site to the security of the other site.

Example: Consider the following `script` tag.

```
<script src="http://www.example.com/js/fancyWidget.js"/>
```

If this tag appears on a web site other than `www.example.com`, then the site is dependent upon `www.example.com` to serve up correct and non-malicious code. If attackers can compromise `www.example.com`, then they can alter the contents of `fancyWidget.js` to subvert the security of the site. They could, for example, add code to `fancyWidget.js` to steal a user's confidential data.

Recommendation

Keep control over the code your web pages invoke. Do not include scripts or other artifacts from thirdparty sites.

5.49 menuHead.jspf

Issue 0BD8D7FD8D723491F349C67C2579C205:

Sink :
File valg_overvaking/src/main/webapp/jspf/menuHead.jspf 6

```
 3|<link rel="stylesheet" href="<%=basePath%>js/jquery/themes/blue/style.css" type="text/css" media="print, 
projection, screen"/>
 4|<link rel="stylesheet" href="<%=basePath%>js/jquery/theme/ui.all.css" type="text/css" media="print, projection, 
screen"/>
 5|<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" type="text/css" 
media="print, projection, screen"/>
>6|<script type="text/javascript" src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
 7|<script type="text/javascript" src="<%=basePath%>js/jquery/jquery.tablesorter-2.22.5.js"></script>
 8|<script type="text/javascript" src="<%=basePath%>js/jquery/jquery.tablesorter.widgets-2.22.5.js"></script>
 9|<script type="text/javascript" src="<%=basePath%>js/jquery/jquery.ui.all.js"></script>
```

Issue 0BD8D7FD8D723491F349C67C2579C206:

Sink :
File valg_overvaking/src/main/webapp/jspf/menuHead.jspf 10

```
  7|<script type="text/javascript" src="<%=basePath%>js/jquery/jquery.tablesorter-2.22.5.js"></script>
  8|<script type="text/javascript" src="<%=basePath%>js/jquery/jquery.tablesorter.widgets-2.22.5.js"></script>
  9|<script type="text/javascript" src="<%=basePath%>js/jquery/jquery.ui.all.js"></script>
>10|<script type="text/javascript" src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
 11|<script type="text/javascript">
 12|    $(document).ready(function () {
 13|        $("#nav-one li").hover(
```

[Low] Poor Error Handling / Overly Broad Catch

Errors

Analysis: Bad Practice

Abstract

The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

Explanation

Multiple catch blocks can get ugly and repetitive, but "condensing" catch blocks by catching a high-level class like `Exception` can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of Java's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention.

Example: The following code excerpt handles three types of exceptions in an identical fashion.

```
  try {
    doExchange();
  }
  catch (IOException e) {
    logger.error("doExchange failed", e);
  }   catch (InvocationTargetException e) {     logger.error("doExchange failed", e);
  }
  catch (SQLException e) {
    logger.error("doExchange failed", e);   }
```

At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows:

```
try {
  doExchange();
}
catch (Exception e) {
  logger.error("doExchange failed", e);   }
```

However, if `doExchange()` is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions derived from `RuntimeException` such as `ClassCastException`, and `NullPointerException`, which is not the programmer's intent.

Recommendation

Do not catch broad exception classes like `Exception`, `Throwable`, `Error`, or <RuntimeException> except at the very top level of the program or thread.

5.50 MainMottak.java

Issue 5B511C4744C52A3162A12579611957B6:

Sink :
File valg_mottak/src/main/java/no/valg/valgnatt/mottak/MainMottak.java 105-108

```
 103|    br.close();
 104|    result = sb.toString();
>105|} catch (Exception e) {
>106|    LOG.warn("Connection failure, " + e.getMessage() + " " + url, e);
>107|    return null;
>108|} finally {
 109|    if (conn != null) {
 110|        conn.disconnect();
```

5.51 MatriseKalkulasjon.java

Issue E77071B137DE8EF4B3398887A7B947C7:

Sink :
File valg_backend/src/main/java/no/valg/valgnatt/backend/mandatberegning/prognose/MatriseKalkulasjon.java 99-105

```
  96|            betas[i] = betaVerdier.get(i, 0);
  97|        }
  98|        return betas;
> 99|    } catch (Exception e) {
>100|        LOG.warn("MÃ¥ bruke apriori beta-verdier, fordi vi har for fÃ¥ kretser til Ã¥ gjÃ¸re
matrisekalkulasjonen: " + e);
>101|        // Dersom vi har for lite data til Ã¥ lÃ¸se matriseligningen, sÃ¥ bruker vi simpelthen apriori-verdiene.
Ettersom flere
>102|        // stemmer blir talt opp, sÃ¥ vil man mer sannsynlig fÃ¥ nok data til Ã¥ lÃ¸se matriselignen, og dermed
fÃ¥ bedre beta-
>103|        // verdier.
>104|        return getAprioriBetaverdier(aprioriBetaverdier);
>105|    }
 106|}
107|
 108|private static double[] getAprioriBetaverdier(double[] aprioriBetaverdier) {
```

5.52 StemmeInnsender.java

Issue 3F86AD0FFCECCDDEF12C0BD43249AD74:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 252-254

```
 249|    System.out.print("Trykk ENTER for Ã¥ fortsette");
 250|    try {
 251|        System.in.read();
>252|    } catch (Exception t) {
>253|        LOG.error("Feil oppstod", t);
>254|     }
 255|}
 256|
 257|public int getAntallInnsendinger() {
```

[Low] Poor Error Handling / Overly Broad Throws

Errors

Analysis: Bad Practice

Abstract

The method throws a generic exception making it harder for callers to do a good job of error handling and recovery.

Explanation

Declaring a method to throw `Exception` or `Throwable` makes it difficult for callers to do good error handling and error recovery. Java's exception mechanism is set up to make it easy for callers to anticipate what can go wrong and write code to handle each specific exceptional circumstance. Declaring that a method throws a generic form of exception defeats this system.

Example: The following method throws three types of exceptions.

```
public void doExchange()
  throws IOException, InvocationTargetException,
         SQLException {
...
}
```

While it might seem tidier to write

```
public void doExchange()
throws Exception {   ...
}
```

doing so hampers the caller's ability to understand and handle the exceptions that occur. Further, if a later revision of `doExchange()` introduces a new type of exception that should be treated differently than previous exceptions, there is no easy way to enforce this requirement.

Recommendation

Do not declare methods to throw `Exception` or `Throwable`. If the exceptions thrown by a method are not recoverable or should not generally be caught by the caller, consider throwing unchecked exceptions rather than checked exceptions. This can be accomplished by implementing exception classes that extend `RuntimeException` or `Error` instead of `Exception`, or add a try/catch wrapper in your method to convert checked exceptions to unchecked exceptions.

5.53 StemmeInnsender.java

Issue A36879ED49DB5315526B3ABB2E7BA89F:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/simulator/StemmeInnsender.java 82-110
Fact Name: no.valg.valgnatt.utils.simulator.StemmeInnsender.utfoer

```
public void utfoer() throws Exception {
    HTTPFasade httpFasade = new HTTPFasade(host);
    if (atomisk) {
        httpFasade.pauseBackend();
    }
    List<Resource> resourcesFiltered = finnStemmeskjemaIKorrektTidsrom(tilfeldig);
    antallInnsendinger = resourcesFiltered.size();

    long forrigeFilTms = 0;
    boolean vent = false;
    long waitedMs = 0;
    String file = null;
    for (int i = 0; i < resourcesFiltered.size(); i++) {
        Resource resource = resourcesFiltered.get(i);
        file = resource.getFilename();
        String ventetid = null;
        if (vent && hastighet != null) {
            ventetid = vent(hastighet, forrigeFilTms, waitedMs, file);
        }
        loggLinjePrRunde(file, i, ventetid);
        waitedMs = send(host, resource.getInputStream());
        forrigeFilTms = tms(filnavnTidspunkt(file));
        vent = true;
    }
    loggLinjeSlutt(file);
    if (atomisk) {
        httpFasade.aktiverBackend();
    }
}

private void loggLinjeSlutt(String file) {
    if ("max".equals(hastighet)) {
```

5.54 Stemmeskjema2Filsystem.java

Issue E1BEBA8A84CC1AB1A787160DAFA08C0D:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/Stemmeskjema2Filsystem.java 57-93
Fact Name: no.valg.valgnatt.utils.Stemmeskjema2Filsystem.printStemmetall

```
>57|private static void printStemmetall(String hovedDir, String db) throws Exception {
>58|    ConfigurationLoader.load(); >59|    try
(Connection c = Database.getConnection(db)) {
>60|        int counter = 0;
>61|        String sql = "SELECT opprettet_tid, valgtype, kommnr, kretsnr, data FROM stemmeskjema";
>62|        try (Statement s = c.createStatement(); ResultSet rs = s.executeQuery(sql)) {
>63|            while (rs.next()) {
>64|                Date opprettet = rs.getTimestamp("opprettet_tid");
>65|                String valgtype = rs.getString("valgtype");
>66|                String kommnr = rs.getString("kommnr");
>67|                String kretsnr = rs.getString("kretsnr");
>68|                String skjema = rs.getString("data");
>69|                File f = new File(hovedDir + File.separator + valgtype + File.separator + kommnr.substring(0, 2) +
File.separator + kommnr);
>70|                if (!f.exists()) {
>71|                    LOG.log(Level.INFO, "Katalogen " + f.getAbsolutePath() + " finnes ikke, mÃ¥ lages");
>72|                    f.mkdirs();
>73|                }
>74|                File file = new File(f.getAbsoluteFile() + File.separator + sdf3.format(opprettet) + "-" + kretsnr
+ ".json");
>75|                if (!file.exists()) {
>76|                    LOG.log(Level.INFO, "File " + file.getAbsoluteFile() + " finnes ikke, mÃ¥ lages");
>77|                    BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file),
"UTF-8"));
>78|                    out.write(toPrettyFormat(skjema));
>79|                    out.flush();
>80|                    out.close();
>81|                }
>82|
>83|                counter++;
>84|                if (counter % HUNDRE == 0) {
>85|                    LOG.log(Level.INFO, "Printed " + counter + " stemmetall");
>86|                }
>87|            }
>88|            LOG.log(Level.INFO, "Printed " + counter + " stemmetall");
>89|        }
>90|    } catch (SQLException | IOException e) {
>91|        throw new Exception(e);
>92|    }
>93|}
94|
 95|private static String toPrettyFormat(String jsonString) {
 96|    JsonObject json = PARSER.parse(jsonString).getAsJsonObject();
```

5.55 Stemmeskjema2Mottak.java

Issue BA8F8F3CE1017200BC7C2FD99F374EF8:

Sink :
File valg_utils/src/main/java/no/valg/valgnatt/utils/Stemmeskjema2Mottak.java 52-76
Fact Name: no.valg.valgnatt.utils.Stemmeskjema2Mottak.kopierStemmeskjema

```
>52|    private static int kopierStemmeskjema(String host, String db, int sisteId) throws Exception {
>53|        ConfigurationLoader.load();
>54|        try (Connection c = Database.getConnection(db)) {
>55|            String sql = "SELECT id, opprettet_tid, valgtype, kommnr, kretsnr, data FROM stemmeskjema WHERE id > "
+ sisteId + " ORDER BY id ASC";
>56|            try (Statement s = c.createStatement(); ResultSet rs = s.executeQuery(sql)) {
>57|                int id = sisteId;
>58|                while (rs.next()) {
>59|                    id = rs.getInt("id");
>60|                    Date opprettet = rs.getTimestamp("opprettet_tid");
>61|                    String valgtype = rs.getString("valgtype");
>62|                    String kommnr = rs.getString("kommnr");
>63|                    String kretsnr = rs.getString("kretsnr");
>64|                    String skjema = rs.getString("data");
>65|                    HTTPFasade fasade = new HTTPFasade(host);
>66|                    fasade.sendStemmer(skjema);
>67|                    System.out.println("Sent stemmeskjema " + id + " (opprettet " + opprettet + ") for " + kommnr
+ "-" + kretsnr + " i valg " + valgtype);
>68|
                }
                sisteId = id;
            }
            return sisteId;
        } catch (SQLException e) {
            throw new Exception(e);
```

Innledning

Valgdirektoratet arbeider kontinuerlig med å forbedre sine systemer, koderevisjoner er en del av denne forbedringsprosessen og gir viktige innspill for bedring av kodekvalitet og økt sikkerhet. Våren 2018 bestilte Valgdirektoratet koderevisjon for EVA applikasjonene EVA Admin, EVA Skanning og EVA Resultat fra en ekstern markedsaktør.

Målsetning

Målsetningen for oppdraget var å få en uavhengig vurdering av kildekode og systemarkitektur for EVA-porteføljen der svakheter belyses langs to akser:

- *Sikkerhet* – Med sikkerhet menes feil eller mangler i applikasjonskode og/eller arkitektur som vil kunne ha konsekvenser for valgresultatet i den forstand at kode og/eller arkitektur åpner for/ikke tilstrekkelig utelukker 3. parts manipulasjon.
- *Kvalitet* – Med kvalitet menes feil eller mangler i kode og/eller arkitektur som er potensielle feilkilder og/eller som reduserer forståelse, testbarhet og vedlikeholbarhet. Typiske kandidater:
    - Kodekompleksitet
    - Kodeorganisering
    - Feil eller uhensiktsmessig bruk av biblioteker / rammeverk etc.

Vurderingene skulle brukes i forbedringsarbeidet av hele porteføljen for EVA

Koderevisjonsprosess

Leverandør og Valgdirektoratet gjennomførte oppstartsmøte med innføring i applikasjoner og domene før leverandør gjorde sin revisjon og analyse. Ved oppdragets avslutning gikk leverandør gjennom funnrapporter med Valgdirektoratet.

Funnene fra leverandør ble så gjennomgått og vurdert av Valgdirektoratet, der Valgdirektoratet gjorde sin vurdering basert på fra alvorlighetsgrad angitt av leverandør. Utfallet av analysen var tredelt:

- Enkelte funn var falske positiver, dvs. at funnene ikke var reelle funn når kontekst for funnene ble vurdert, disse ble avvist
- De mest alvorlige funnene ble prioritert og rettet før produksjonssetting
- Mindre alvorlige funn ble prioritert ned og inngår i den generelle forbedringsprosessen for EVA produktene. Dette er funn som ikke er direkte sårbarheter, og som vurderes og prioriteres i videre utviklingsløp.

Valgdirektoratet er med denne prosessen trygg på at EVA-applikasjonene holder god kvalitet og er tilstrekkelig sikret for valget i 2019.

Avviksrapport – EVA Resultat

Anbefalinger – Kodekvalitet og arkitektur

Anbefaling 1 Se over eventuelle avhengigheter mellom applikasjons- og testkode

Vurderes og gjøres evt. i sammenheng med videre utvikling.

Anbefaling 2 Bruk samme versjon for alle komponenter innenfor samme biblioteksgruppe

Vurderes og gjøres evt. i sammenheng med videre utvikling.

Verktøybaserte funn

Komponenter med sårbarheter

Sårbarhetene beskrevet i rapporten er utbedret der nyere versjon finnes, eller der oppgradering til nyere versjon ikke hindres (se reelle funn under).

Valgdirektoratet har automatisert owasp i CI/CD byggpipeline som anbefalt av Mnemonic. Owasp-funn vurderes løpende i utviklingsprosessen som følger:

- Falske positiver / ikke direkte sårbarheter: Sårbarheter i biblioteker som brukes av biblioteker som brukes av EVA Resultat, men der underliggende biblioteks funksjonalitet ikke brukes
- Reelle funn:
  - Oppgraderes direkte dersom det finnes nyere versjon tilgjengelig og dersom konsekvensene av oppgradering er kontrollerbare
  - Planlegges dersom det finnes nyere versjon tilgjengelig, men der konsekvensene av oppgradering ikke er kontrollerbare
  - Legges på vent dersom nyere versjon ikke er tilgjengelig

Verktøybaserte funn

Mange av avvikene som ble funnet i EVA Resultat er avvik i Valgdirektoratets interne overvåkingsverktøy som brukes i forbindelse med prognoseberegningen på valgnatten, dette gjør at Valgdirektoratet har vurdert kritikaliteten som lav og således ikke har prioritert å utbedre avvikene.

Den samme vurderingen er gjort for EVA Resultat siden denne applikasjonen heller ikke er eksponert eller tilgjengelig for andre enn Valgdirektoratet, med unntak av frontend/API-tjenesten, se systemdokumentasjonen for nærmere forklaring.

Valgdirektoratet vil på sikt etterstrebe å utbedre funnene for på den måten å bedre etterleve prinsippet om sikkerhet gjennom lagdeling, samt å følge «best practice» i implementasjon.

Kildekodefunn

Avvikslogg – Utbedrede funn

Avvik	Kommentar
2EE Misconfiguration / Missing Error Handling	Utbedret

Avvikslogg – False positiver / ikke relevante funn.

Avvik	Kommentar
Cross-Site Scripting / Persistent Input Validation and Representation, Analysis: Suspicious	Dette er et internt overvåkningsverktøy

Cross-Site Scripting / Reflected Input Validation and Representation, Analysis: Suspicious	Dette er et internt overvåkningsverktøy
Password Management / Empty Password in Configuration File Environment, Analysis: Suspicious	Falsk positiv. Det er ikke tomt passord som er passordet. Passordet er ikke oppgitt i fila.
Password Management / Password in Configuration File Environment, Analysis: Not an Issue / Suspicious	Dette er konfigurasjon til bruk under utvikling på lokal maskin.
Privacy Violation / Autocomplete Security Features, Analysis: Suspicious	Dette er et internt overvåkningsverktøy
Resource Injection Input Validation and Representation, Analysis: Not an Issue / Suspicious	Dette er et internt overvåkningsverktøy
Server-Side Request Forgery Input Validation and Representation, Analysis: Suspicious	Dette er et internt overvåkningsverktøy
Session Fixation Time and State, Analysis: Suspicious	Dette er et internt overvåkningsverktøy
Cross-Site Request Forgery	Dette er et internt overvåkningsverktøy

Analysis: Suspicious Abstract: Form posts must contain a user-specific secret in order to prevent an attacker from making unauthorized requests.	

Avvikslogg – Restanseliste mindre kritiske funn

Avvik	Kommentar
Race Condition / Format Flaw Analysis: Reliability Issue Abstract: The methods parse() and format() in java.text.Format contain a design flaw that can cause one user to see another user's data.	Det er ingen brukere i systemet - backlog
Denial of Service / StringBuilder Analysis: Not an Issue / Reliability Issue Abstract: Appending untrusted data to StringBuilder instance initialized with the default backing-array size can cause the JVM to over-consume heap memory space.	Backlog
J2EE Misconfiguration / Missing Data Transport Constraint Analysis: Suspicious Abstract: A security constraint that does not specify a user data constraint cannot guarantee that restricted resources will be protected at the transport layer.	Backlog

Missing Check against Null Analysis: Reliability Issue Abstract: The program can dereference a null pointer bedcause it does not check the return value of a function that might return null.	Backlog
Missing Check for Null Parameter Analysis: Reliability Issue Abstract: This function violates the contract that it must compare its parameter with null.	Backlog
Poor Error Handling / Throw Inside Finally Analysis: Bad Practice Abstract: Using a throw statement inside a finally block breaks the logical progression through the try-catch-finally.	Backlog
Redundant Null Check Analysis: Reliability Issue Abstract: The program can dereference a null pointer, thereby causing a null pointer exception.	Backlog
System Information Leak / Incomplete Servlet Error Handling Analysis: Suspicious Abstract: If a Servlet fails to catch all exceptions, it might reveal debugging information that will help an adversary form a plan of attack.	Backlog

Unreleased Resource / Streams Analysis: Reliability Issue Abstract: The program can potentially fail to release a system resource.	Backlog
Denial of Service Analysis: Reliability Issue Abstract: An attacker could cause the program to crash or otherwise become unavailable to legitimate users.	Backlog
Hardcoded Domain in HTML Analysis: Reliability Issue Abstract: Including a script from another domain means that the security of this web page is dependent on the security of the other domain.	Backlog
Poor Error Handling / Overly Broad Catch Analysis: Bad Practice Abstract: The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.	Backlog
Poor Error Handling / Overly Broad Throws Analysis: Bad Practice	Backlog

Abstract: The method throws a generic exception making it harder for callers to do a good job of error handling and recovery.	