3/2/22

Project 2 Bug Analysis and Mitigation Report

PREVENTING BUGS MOVING FORWARDS

MARQUIS, ALEX

Table of Contents

Identified Bugs and Vulnerabilities

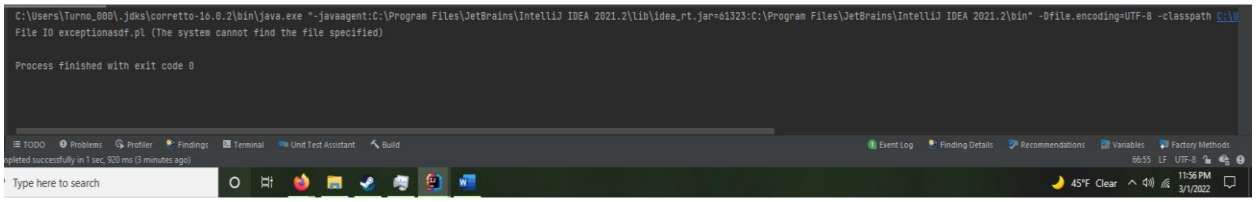
The bugs identified in the bug report seen in the appendix was generated by Douglas Lundin on January 28th, 2022, were violations of various recommendations from Carnegie Mellon University's Software Engineering Institute's (SEI) CERT division. The mitigation of the bugs documented in this report was assigned to Alexander Marquis on February 13th and mitigations were approved on March 2nd, 2022.

(PR2-ERR01-01) Exposed Sensitive File System Information Through IO Exceptions

Bug PR2-ERR01-01 stems from a failure to follow SEI CERT recommendation *ERR01-J. Do not allow exceptions to expose sensitive information*[1]. Failure to follow this recommendation has created conditions where IO exceptions suffered in the first try block of the original source code have the potential to cause leaks of sensitive file system information during exception handling.

```
String filename = args[0];
BufferedReader inputStream = null;

String fileLine;
try {
    inputStream = new BufferedReader(new FileReader(filename));
```

As demonstrated in the offending code above, taken from lines 9-14 of the original source code, the actual IO exception message is printed directly to the console. Printing the IO exception message to the console risks exposing sensitive information about the file system through information gleamed from the exception messages[1]. The bug itself can be demonstrated by loading a faulty or nonexistent filename as the program argument and executing the program.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61323:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
File IO exceptionasdf.pl (The system cannot find the file specified)

Process finished with exit code 0
```

The output from the program execution above, where the inputted filename is *asdf.pl*, demonstrates leakage of sensitive file system information through poor exception handling. The user should not be able to know what files do and do not exist within the secured directory from output generated by exceptions.

[1] Mohindra, Dhruv, et al. "ERR01-J. Do Not Allow Exceptions to Expose Sensitive Information." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/ERR01-J.+Do+not+allow+exceptions+to+expose+sensitive+information.

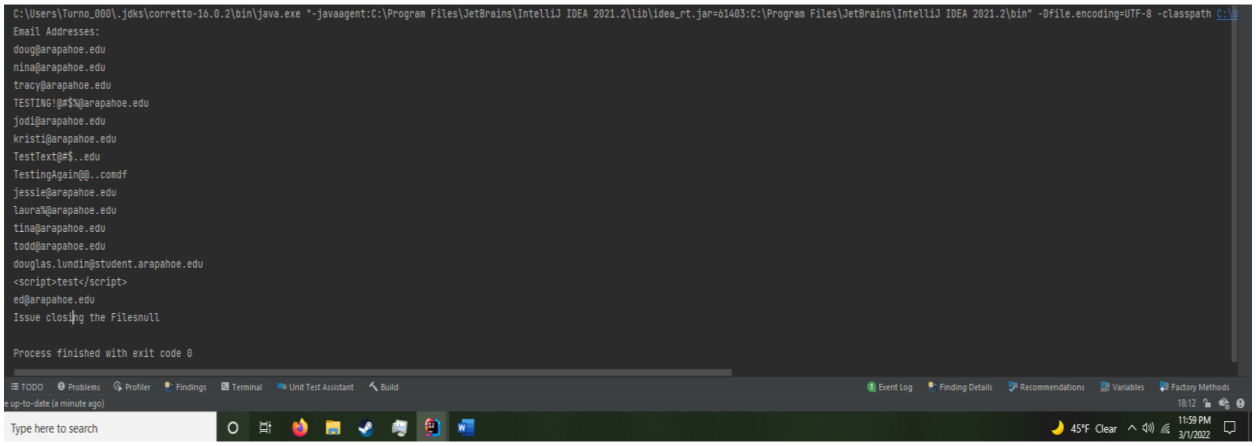
(PR2-ERR01-02) Exposed Failure to Close File Through IO Exceptions

Bug PR2-ERR01-01 stems from a failure to follow SEI CERT recommendation *ERR01-J. Do not allow exceptions to expose sensitive information*[1]. Failure to follow this recommendation has created conditions where IO exceptions suffered while attempting to close a BufferedReader in the second try block of the original source code have the potential to cause leaks of sensitive file system information during exception handling.

```
} catch (IOException io) {
    System.out.println("Issue closing the Files" + io.getMessage());
}
```

The block of code above, taken from lines 30-32 of the original source code, demonstrates the program both informing the user of the file failing to close and printing the actual IO exception message. Printing the IO exception message can lead to unintentional exposure of sensitive file system information[1]. This practice allows malicious individuals to gain more knowledge about the software and the systems it runs on which in turn allows them a better opportunity at exploiting the vulnerabilities present within the system. The bug itself can be demonstrated by temporarily implementing the statement shown below into the try block that is connected to the catch block above.

```
throw new IOException();
```



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61403:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
Email Addresses:
doug@arapahoe.edu
nina@arapahoe.edu
tracy@arapahoe.edu
TESTING!@#$%@arapahoe.edu
jodi@arapahoe.edu
kristi@arapahoe.edu
TestText@#$..edu
TestingAgain@@..comdf
jessie@arapahoe.edu
laura%@arapahoe.edu
tina@arapahoe.edu
todd@arapahoe.edu
douglas.lundin@student.arapahoe.edu
<script>test</script>
ed@arapahoe.edu
Issue closing the Filesnull

Process finished with exit code 0
```

The above output from the program execution, where *Email_addresses_20210205.txt* is the program argument and the *throw new IOException();* statement shown above is implemented demonstrates leakage of sensitive file system information, specifically the failure of a file to properly close. This information should be restricted from users and securely logged for future analysis.

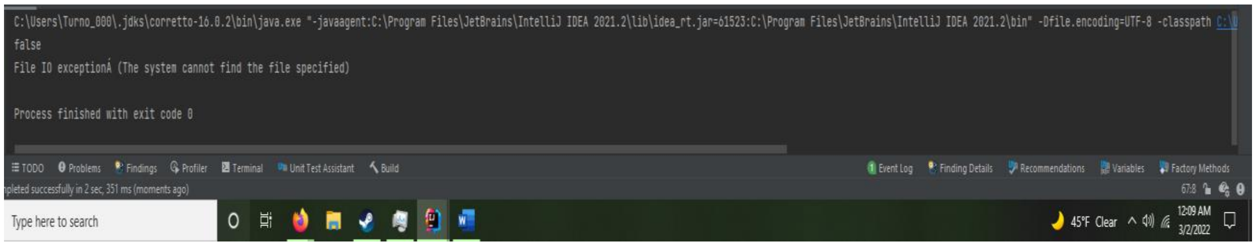
(PR2-IDS01-01) Failure to Normalize Program Argument Before Validation

Bug PR2-IDS01-01 stems from a failure to follow SEI CERT recommendation *IDS01-J. Normalize strings before validating them*[2]. Failure to follow this recommendation has resulted in conditions that allow for several types of exploitation and generates a poor user experience through confusion and frustration. The confusion and frustration stems from what appear to be the same characters not having the same character codes, so they won't match when compared against each other.

```
String filename = args[0];
BufferedReader inputStream = null;

String fileLine;
try {
    inputStream = new BufferedReader(new FileReader(filename));
```

The offending block of code that was taken from lines 9-14 of the original source code demonstrate that no normalization is occurring to the inputted filename before it is used to instantiate a BufferedReader to read a file's contents. Validation is also not occurring though it is addressed further in depth in bug *PR2-IDS51-01: Failure to Enforce Conservative File Naming Conventions*. The bug can be demonstrated by implementing the statement found below and loading the A-acute character (Á) as the program argument.

```
String filename = args[0];
System.out.println(String.valueOf(filename).equals("\u0041\u0301"));
```



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61523:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\
false
File IO exceptionÁ (The system cannot find the file specified)

Process finished with exit code 0
```

The program output above, specifically the line that reads false, informs us that the A-acute character is not being normalized into its decomposed form. Decomposing filenames allows for matches for filenames to be more effectively found and more readily stripped of characters that don't meet the file naming convention.

[2] Mohindra, Dhruv, et al. "IDS01-J. Normalize Strings before Validating Them ." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS01-J.+Normalize+strings+before+validating+them.

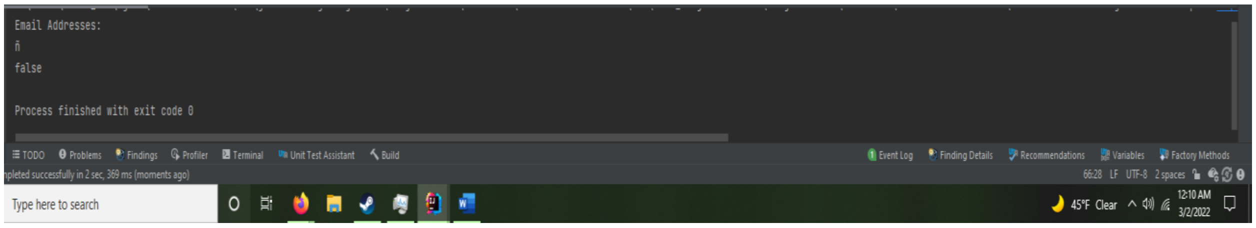
(PR2-IDS01-02) Failure to Normalize File Contents Before Validation

Bug PR2-IDS01-01 stems from a failure to follow SEI CERT recommendation *IDS01-J. Normalize strings before validating them*[2]. Not following this recommendation has resulted in conditions that allow for malicious code execution when the output is redirected from the console to another program.

```
while ((fileLine = inputStream.readLine()) != null) {
    System.out.println(fileLine);
}
```

The lines of code displayed above are the offending lines that result in this bug existing. It occupies lines 18-21 and clearly demonstrates a lack of normalization happening to the inputted contents of files. The contents of files are read line by line and directly outputted by the print statement to the console. A lack of normalization means that when bug PR2-IDS51-01 is encountered the encoding and escaping of output would be even more difficult as it has not been normalized. The bug can be demonstrated by creating a new .txt file containing only the character ñ. Then the statement seen below is added directly below the print statement seen in the code block above.

```
System.out.println(String.valueOf(fileLine).equals("\u00f1"));
```



```
Email Addresses:
ñ
false

Process finished with exit code 0
```

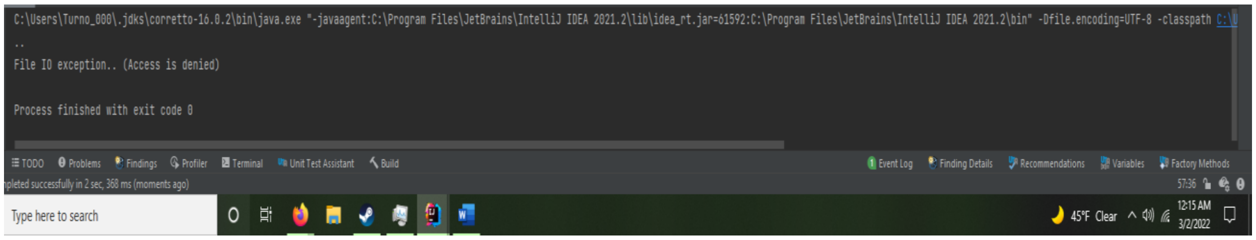
The program output above demonstrates that the ñ is not being normalized to its NFKC form. Since multiple Unicode encodings of the ñ exists, the demonstration that they are not all being modified to have the same encoding effectively demonstrates a lack of normalization in the output.

(PR2-FIO16-01) Failure to Normalize Program Argument Before Validation

Bug PR2-FIO16-01 stems from a failure to follow SEI CERT recommendation *FIO16-J. Canonicalize path names before validating them*[3]. Failure to follow this recommendation has resulted in conditions that allow for several types of exploitations the most obvious of which being path transversal exploitations.

```
String filename = args[0];
BufferedReader inputStream = null;

String fileLine;
try {
    inputStream = new BufferedReader(new FileReader(filename));
```

As the source code that is displayed above demonstrates, no canonicalization is occurring before the path is used to instantiate a BufferedReader to read from whatever file the path points to. The offending block of code, taken from lines 9-14 of the original source code, also lacks any kind of validation of the file path to ensure that canonicalization has succeeded and that the path can only point to files within a secure, whitelisted directory. The bug can be demonstrated by loading two periods as the program argument and executing the program.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61592:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
..
File IO exception.. (Access is denied)

Process finished with exit code 0
```

The program output shown above demonstrates that canonicalization and validation are not occurring. The file should not be able to have a filename of just two periods as that is symbolic of ascending to the parent directory in the file system. The fact that it is printing out the exception information is both part of bug *PR2-ERR01-01: Exposed Sensitive File System Information Through IO Exceptions* and represents that validation is not occurring and the file is not being accessed due to an IO exception which is an undesirable standard for whether a file is secure to access.

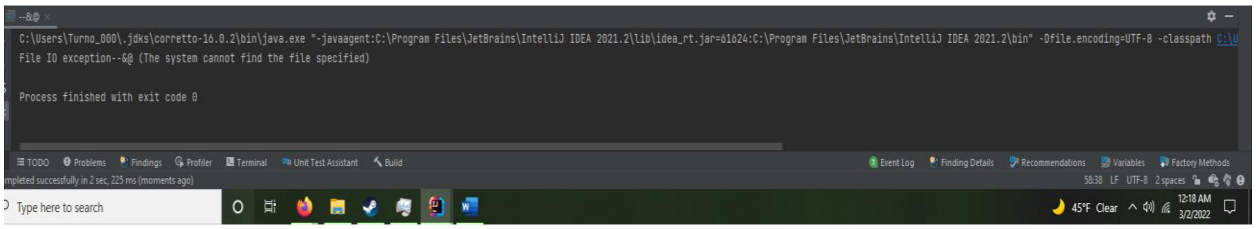
[3] Mohindra, Dhruv, et al. “FIO16-J. Canonicalize Path Names before Validating Them .” *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/FIO16-J.+Canonicalize+path+names+before+validating+them.

(PR2-IDS50-01) Failure to Enforce Conservative File Naming Conventions

Bug PR2-IDS50-01 stems from a failure to follow SEI CERT recommendation *IDS50-J. Use conservative file naming conventions*[4]. Failure to follow this recommendation has allowed for conditions where potentially dangerous character sequences are accepted as valid file names and are used as such during program operation. Beyond dangerous character sequences it also allows for confusing and not helpful filenames like --&@.

```
String filename = args[0];
BufferedReader inputStream = null;

String fileLine;
try {
    inputStream = new BufferedReader(new FileReader(filename));
```

These lines of code, taken from lines 9-14 of the original source code, demonstrate that no form of filename validation is occurring before the filename is operated upon and used as a path by the BufferedReader/FileReader objects. This bug can be demonstrated by feeding in any number of bad file names but for the example shown below the program argument *--&@* was used.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61624:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
File IO exception--&@ (The system cannot find the file specified)

Process finished with exit code 0
```

The program output above demonstrates that potentially malicious filenames such as those beginning with double hyphens are not validated to ensure that they don't get processed. It also allows for a broad character set to be included in the file names which again introduces sources of vulnerabilities.

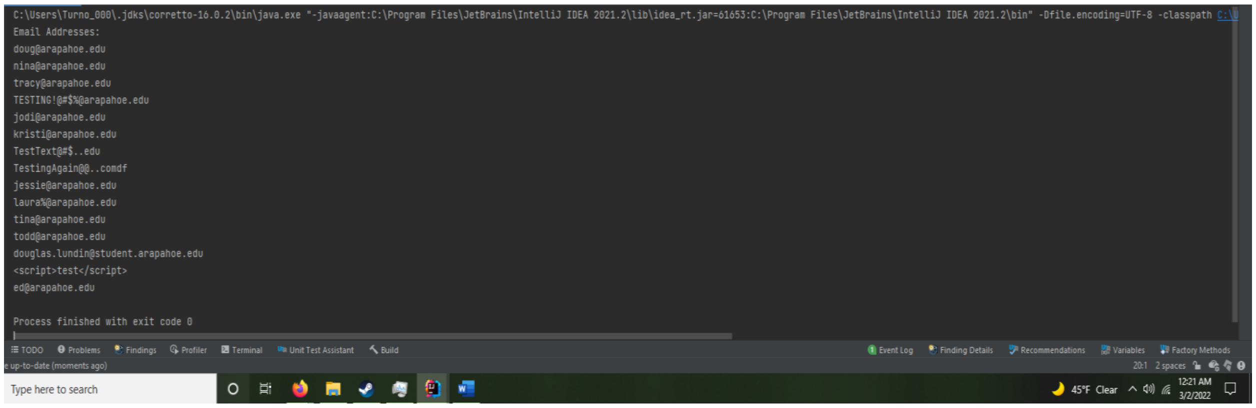
[4] Svoboda, David, et al. "IDS50-J. Use Conservative File Naming Conventions." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS50-J.+Use+conservative+file+naming+conventions.

(PR2-IDS51-01) Failure to Properly Encode or Escape Output of File Contents

Bug PR2-IDS51-01 stems from a failure to follow SEI CERT recommendation *IDS51-J. Properly encode or escape output*[5]. Failure to follow this recommendation has allowed for conditions where potentially dangerous character sequences are outputted. While the console is usually a relatively safe environment for text output since many platforms allow for the redirection of the output this bug must considered.

```java
while ((fileLine = inputStream.readLine()) != null) {
    System.out.println(fileLine);
}
```

The block of code above, taken from lines 18-21 of the original source code, demonstrates that no encoding or escaping of the file's contents is occurring prior to output. The file's contents are directly read line by line in a while loop, and then on each iteration the line is directly outputted. This bug can be demonstrated directly by simply executing the program using the standard program argument of *Email_addresses_20210205.txt* and observing the output.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61653:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
Email Addresses:
doug@arapahoe.edu
nina@arapahoe.edu
tracy@arapahoe.edu
TESTING!@#$%@arapahoe.edu
jodi@arapahoe.edu
kristi@arapahoe.edu
TestText@#$..edu
TestingAgain@@..comdf
jessie@arapahoe.edu
laura%@arapahoe.edu
tina@arapahoe.edu
todd@arapahoe.edu
douglas.lundin@student.arapahoe.edu
<script>test</script>
ed@arapahoe.edu

Process finished with exit code 0
```

As the program output above demonstrates, potentially dangerous character sequences are being directly outputted with no encoding or escaping such as the script tags present in the output. Any malicious input present in the file will be directly outputted by the program.

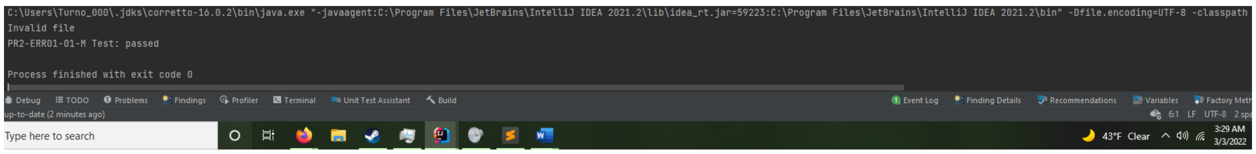
[5] Mohindra, Dhruv, et al. "IDS51-J. Properly Encode or Escape Output." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS51-J.+Properly+encode+or+escape+output.

Implemented Mitigations

All mitigations discussed were implemented by Alexander Marquis and approved on March 2nd, 2022. These are direct mitigations to the bugs documented in the bug report produced by Douglas Lundin and that were displayed in the previous section. All mitigations have an ID and take the form of the bug ID that they are mitigating with a -M added to the end representing that this is a mitigation to that particular bug.

(PR2-ERR01-01-M) Secure Filesystem Information When Handling IO Exceptions

Mitigation PR2-ERR01-01-M is a mitigation to bug PR2-ERR01-01. This mitigation is simple and required little time or resources to implement. The output of the print statement inside the catch block that held the bug this mitigation goes to was modified to simply read “Invalid file”. This secures filesystem information as it conceals the reason the file is considered invalid whereas originally it would explicitly state why the file failed to validate when IO exceptions occurred which had the potential to leak sensitive filesystem information. This mitigation complies with SEI CERT recommendation *ERR01-J. Do not allow exceptions to expose sensitive information*[1].

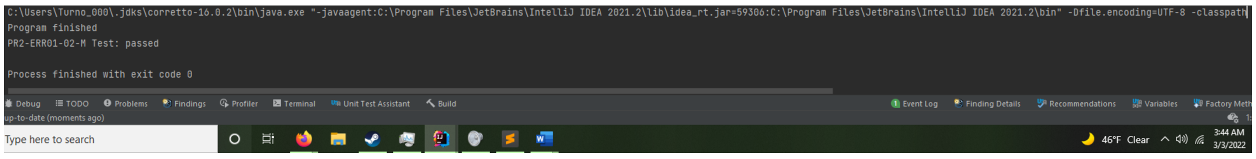


```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=59223:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath
Invalid file
PR2-ERR01-01-M Test: passed

Process finished with exit code 0
```

The output above comes from running the *pr2err0101mTest* method instead of the primary start method in the final mitigated code. This output helps to demonstrate that mitigation PR2-ERR01-01-M is effective by simulating the same actions performed during the standard implementation of the mitigation. The try-catch block surrounding the print statement is replicated. Then, an intentional IO exception is thrown at the top of the try block as to purposefully trigger the catch block to see what happens when the program suffers a general IO exception. The result is clear, when general IO exceptions are suffered, the program will inform the user that the chosen file was invalid.

(PR2-ERR01-02-M) Secure File Open Status When Handling IO Exceptions

Mitigation PR2-ERR01-02-M is a mitigation to bug PR2-ERR01-02. This mitigation is simple and required little time or resources to implement. The output of the print statement inside the catch block that held the bug this mitigation goes to was modified to simply read “Program finished”. The exception is logged so that it can later be analyzed. These steps secure sensitive filesystem information when handling IO exceptions caused by files being unable to close properly as they don’t inform the user that the reason the program finished was because it was unable to properly close a file. This mitigation complies with SEI CERT recommendation *ERR01-J. Do not allow exceptions to expose sensitive information*[1].

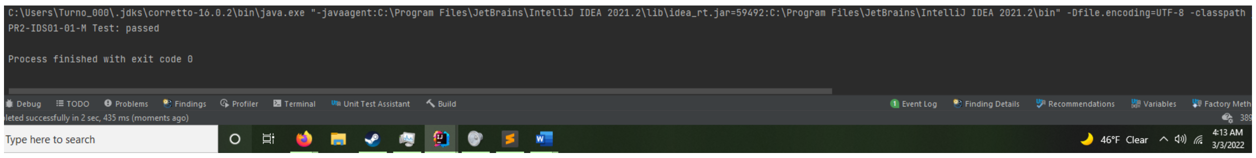


```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=59306:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath
Program finished
PR2-ERR01-02-M Test: passed

Process finished with exit code 0
```

The above output comes as a result of running the *pr2err0102mTest* method instead of the primary start method in the final mitigated code. This output helps to demonstrate that mitigation PR2-ERR01-02-M is effective by simulating the same actions performed during the standard implementation of the mitigation. The try-catch block that surrounds the print statement that was affected by this bug is replicated inside the test method. At the top of the replicated a try block, an intentional IO exception is thrown to trigger the catch block containing the mitigation. Since the mitigation no longer reveals sensitive filesystem information by simply stating that the program has finished it effectively mitigates the problem.

(PR2-IDS01-01-M) Normalization of Program Argument Before Validation

Mitigation PR2-IDS01-01-M is a mitigation to bug PR2-IDS01-01. This mitigation was a low-cost mitigation as well with it taking little time or code to rectify. The mitigation is half of the *normalizeAndSanitize* method. In this method, a file name is normalized to its canonically decomposed form, otherwise known as the NFD form. The filenames are then sanitized of all characters that are not alphanumeric, periods (.), underscores (_), and hyphens (-). On top of the standard benefits of normalization like having the same character that has multiple Unicode encodings be recognized and matched as being the same character, this normalization before sanitization allows for the sanitizing method to save as many of the alphanumeric characters as it can by stripping their accents off them and writing them in their canonically decomposed forms. This mitigation complies with SEI CERT recommendation *IDS01-J. Normalize strings before validating them*[2].



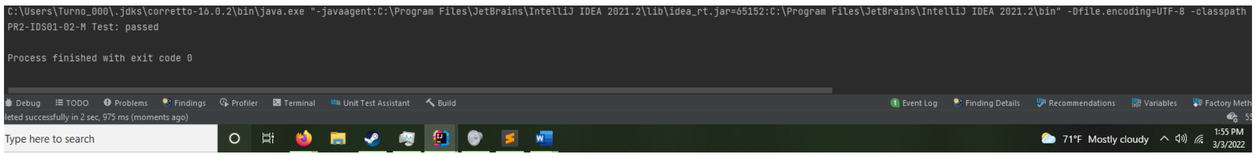
```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=59492:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath
PR2-IDS01-01-M Test: passed

Process finished with exit code 0
```

The program output above is the result of running the *pr2ids0101mTest* method instead of the primary start method in the mitigated code. This method functions by taking several strings with known expected outputs and comparing them with their expected outputs to ensure that proper normalization and sanitization of filenames has occurred. If normalization is working properly, then most accents will be stripped off characters and they will be sanitized back to their base Latin form. The used inputs, expected outputs, and actual outputs of each string evaluated by the method can be seen in the table below.

pr2ids0101mTest Method Inputs & Outputs		
Input	Expected Output	Actual Output
Áslt	Aslt	Aslt
ñÅö	nAo	nAo
ḱṷố	kuo	kuo
bad-File-Name!	bad-File-Name	bad-File-Name
Still%Bad	StillBad	StillBad
Wo&^rs3+=filen{}[]|\$@*?,/ame	Wors3filename	Wors3filename
\u0041\u0301	A	A

(PR2-IDS01-02-M) Normalization of File Contents Before Validation

Mitigation PR2-IDS01-01-M is a mitigation to bug PR2-IDS01-01. This mitigation, much like the other normalization mitigation, was a simple and low-cost to implement as it merely involved using Java's Normalier.normalize method from its standard library. This mitigation was placed inside the newly added *printFileContents* method and occurs to every line read before it is encoded into an HTML safe format as part of mitigation PR2-IDS51-01-M. This mitigation complies with SEI CERT recommendation *IDS01-J. Normalize strings before validating them*[2].



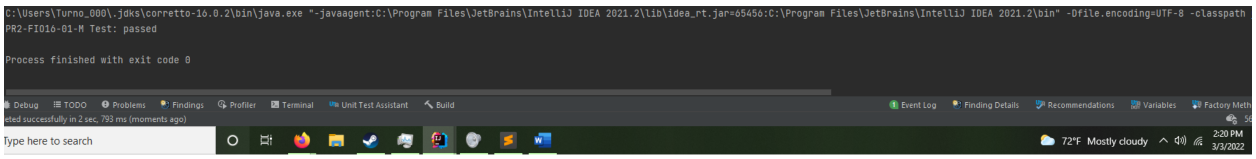
```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=65152:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath
PR2-IDS01-02-M Test: passed

Process finished with exit code 0
```

The output above is from running the *pr2ids0102mTest* method instead of the primary start method in the mitigated code. This method functions by calling the normalization method mentioned above on a series of strings with expected outputs after normalization in the NFKC form. These normalized strings are then compared against their expected outputs. If any one of the strings fail to match its expected output, then the test will fail representing that normalization has not reached the desired output. The used inputs, expected outputs, and actual outputs of each string tested by the method can be seen in the table below.

pr2ids0102mTest Method Inputs & Outputs		
Input	Expected Output	Actual Output
\u006e\u0303	\u00f1	\u00f1
ñ	\u00f1	\u00f1
ñ	\u00f1	\u00f1
\u00f1	\u00f1	\u00f1
-.!()	\u002d\u002e\u0021\u0028	\u002d\u002e\u0021\u0028

(PR2-FIO16-01-M) Canonicalization of File Path Before Validation

Mitigation PR2FIO16-01-M is a mitigation to bug PR2-FIO16-01. This mitigation took a little more time than other mitigations as the validation for path names is more involved. First, the whitelisted directory was moved to a secure location outside the application server and the path to the directory is now accessed via an environment variable. Next the whitelisted path is accessed from that environment variable and used as the preface to the inputted filename to generate a canonical path. The generation of the canonical path comes from Java's *File.getCanonicalPath* method which will get the canonical path to a file of a given path. Finally, after the path has been canonicalized it is then verified to ensure that it points to a file within the whitelisted directory. This validation is accomplished by both taking the parent directory of the canonicalized path and comparing it against the secure, whitelisted directory as they should match and, if that stage of validation passes, then the canonicalized path is compared against each of the children files from the secure, whitelisted directory. Only if the canonicalized path matches the path of one of the child files from the secure whitelisted directory is the canonicalized path considered validated. This mitigation complies with SEI CERT recommendation *FIO16-J. Canonicalize path names before validating them*[3]



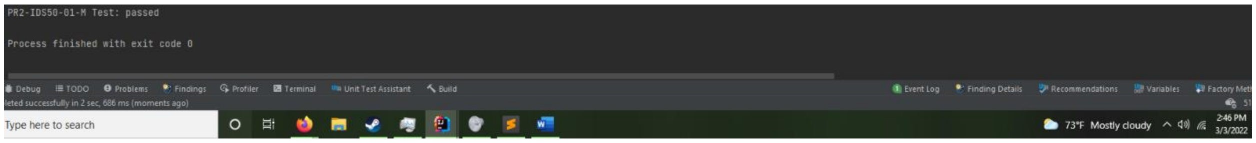
```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=65456:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath
PR2-FIO16-01-M Test: passed

Process finished with exit code 0
```

The program output above is the result of running the *pr2fio1601mTest* method instead of the primary start method in the mitigated code. The *pr2fio1601mTest* method works by taking a series of strings representing possible filenames, canonicalizing them using the secure, whitelisted directory's path as the base, it then calls the validation method used above for validating file paths and compares the output of the validation against the expected output for the validation. If any one of the validations doesn't match its expected output, the whole test will fail indicating that the validation method does not have the expected output. However, this test passes indicating that the mitigation is achieving its desired goal.

pr2fio1601mTest Method Inputs & Outputs		
Input	Expected Output After Validation	Actual Output After Validation
Email_addresses_20210205.txt	true	true
goodFileNameButNotReal	false	false
%bad!file^name	false	false
".." + File.separator + "README.md"	false	false
".." + File.separator + "Email_addresses_20210205.txt"	false	false
\\\\\\\	false	false

(PR2-IDS50-01-M) Enforcement of Conservative File Naming Conventions

Mitigation PR2-IDS50-01-M is a mitigation to bug PR2-IDS50-01. This mitigation took quite a while to implement as a conservative file naming convention had to be established. The established file naming convention has the following requirements for file names. Filenames must be at least 2 characters long and must be no longer than 255 characters in length. The first and last characters of a filename must be alphanumeric. Filenames may only contain alphanumeric characters, periods (.), underscores (_), and hyphens (-). Finally, periods and hyphens may not appear one after another, meaning there cannot be two periods or hyphens next to each other and there also may not be a period and a hyphen next to each other. All these requirements are enforced by a single validation regex except for the maximum size constraint which is the first thing checked and it is validated using a simple Java string length check. All these requirements are enforced in the *validateFilename* method that is called and must return true before any further operations take place using the filename such as canonicalization. This mitigation complies with SEI CERT recommendation *IDS50-J. Use conservative file naming conventions*.



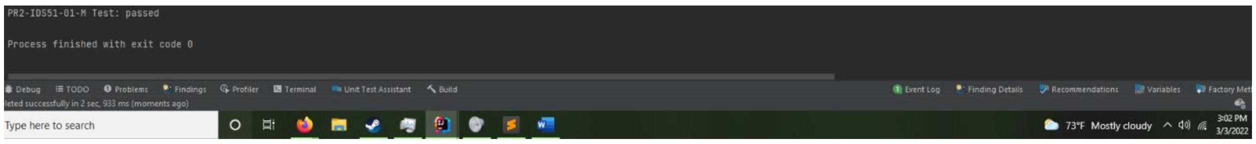
```
PR2-IDS50-01-M Test: passed

Process finished with exit code 0
```

The program output above is the result of running the *pr2ids5001mTest* method instead of the primary start method in the mitigated code. The *pr2ids5001mTest* method works by taking several strings representing inputted file names, using the validation method on those inputted strings, and comparing the result of the validation method against the expected result of the validation. If any one of the validations doesn't match its expected result the whole test fails indicating that the validation method fails to securely validate only files that adhere to the established file naming convention. Only if all validations match their expected results will the test pass. The table displaying the inputs, expected outputs after validation, and actual outputs after validation for the *pr2ids5001mTest* Method can be found on the next page

pr2ids5001mTest Method Inputs & Outputs		
Input	Expected Output After Validation	Actual Output After Validation
test-file	true	true
test.file	true	true
testFile	true	true
-test-file	false	false
.test-file	false	false
test-file.	false	false
test-file-",	false	false
test..file	false	false
test--file	false	false
test!file	false	false
test^file	false	false
\ttestFile	false	false
(testfile	false	false
&%^@terset.pl	false	false
""	false	false
256 “a” characters	false	false

(PR2-IDS51-01-M) Encoding and Sanitizing the Output of File Contents

Mitigation PR2-IDS51-01-M is a mitigation to bug PR2-IDS51-01. This mitigation took the longest to implement as there were many considerations. The console is generally a secure environment and printing text to the console is relatively secure to do. However, the output from a console can often be redirected into a program where that output becomes dangerous. After careful consideration, it was decided that the attack vector that was most important to protect against was malicious HTML output. Hence why the mitigation for this bug was to encode all non-alphanumeric characters in the output into their HTML safe formats. This was implemented in the form of a *HTMLEntityEncode* method that was directly taken from the compliant solution for SEI CERT code IDS50-J (Mohindra, Dhruv, et al. *IDS51-J. Properly encode or escape output*). This ensures that no tags, such as malicious script tags, can be passed through to the output in a condition that is dangerous. This mitigation complies with SEI CERT recommendation *IDS51-J. Properly encode or escape output*.



```
PR2-IDS51-01-M Test: passed

Process finished with exit code 0
```

The output above is the result of running the *pr2ids5101mTest* method instead of the primary start method in the mitigated code. The *pr2ids5101mTest* method tests that the mitigation PR2-IDS51-01-M works by taking several inputs that have expected outputs after being properly encoded into a HTML safe format, encoding them using the *HTMLEntityEncode* method used in the mitigation, and testing the output of that encoding against its expected out. If any one of the outputs fails to match its expected output, the whole test fails indicating that mitigation PR2-IDS51-01-M does not achieve the desired goal of properly encoding output into a HTML safe format. The table displaying the inputs, expected outputs after validation, and actual outputs after validation for the *pr2ids5101mTest* Method can be found on the next page

pr2ids5101mTest Method Inputs & Outputs		
Input	Expected Output	Actual Output
!@#$%^&*()-=;:'\",.<>/?|\\[]{}	&#33;&#64;&#35;&#36;&#37;&#94;&#38;&#42;&#40;&#41;&#45;&#61;&#59;&#58;&#39;&#34;&#44;&#46;&#60;&#62;&#47;&#63;&#124;&#92;&#91;&#93;&#123;&#125;"	&#33;&#64;&#35;&#36;&#37;&#94;&#38;&#42;&#40;&#41;&#45;&#61;&#59;&#58;&#39;&#34;&#44;&#46;&#60;&#62;&#47;&#63;&#124;&#92;&#91;&#93;&#123;&#125;"
\uD835\uDCF7\uD835\uDD93	nn	nn
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiumod tempor incididunt	Lorem ipsum dolor sit amet&#44; consectetur adipiscing elit&#44; sed do eiumod tempor incididunt	Lorem ipsum dolor sit amet&#44; consectetur adipiscing elit&#44; sed do eiumod tempor incididunt
<script>Malicious-JavaScript-Code<\script>	&#60;script&#62;Malicious&#45;JavaScript&#45;Code&#60;&#92;script&#62;	&#60;script&#62;Malicious&#45;JavaScript&#45;Code&#60;&#92;script&#62;

Recommendations Moving Forwards

The mitigation of the bugs found in this report took time and resources that could've been spent improving the software or workflow. Having to go back into the code to mitigate bugs is an arduous process that can cost the organization much over time. Instead the focus moving forwards should be to prevent the implementation of bugs in the first place. The following recommendations are made with the intent to decrease the number of implemented bugs into the future and to save on time and resources spent mitigating bugs that could've been simply avoided to begin with.

Adhere to SEI CERT Recommendations

Adherence to SEI CERT recommendations is an effective way to prevent bugs from appearing into the future. It is important to note that these are merely recommendations from SEI CERT and not rules. These can not always be applied to the software and application of the recommendations may rarely have negative consequences. In general, however, following SEI CERT recommendations decreases the likelihood of introducing bugs or vulnerabilities into the software. Adherence to the SEI CERT recommendations could've prevented all bugs detected in the bug report generated by Douglas Lundin on January 28th, 2022. Not following SEI CERT recommendations should always be done with extreme caution and with good justification as to why the SEI CERT recommendation shouldn't be used in this case. In all cases where good justification does not exist for not following SEI CERT recommendation they should be followed.

Continuation of Established File Naming Convention

In mitigation PR2-IDS50-01-M a conservative file naming convention is established in order to process inputted filenames. It is my recommendation that then conservative file naming convention established in mitigation PR2-IDS50-01-M be used and continued into the future. The file naming convention establishes that all file names must be between 2-255 (inclusive) in length. They must have alphanumeric characters as the starting and ending characters and the only characters allowed throughout a file name are alphanumeric characters, periods (.), underscores (_), and hyphens (-). Finally, periods and hyphens may not appear one after another meaning that two periods or hyphens may occur one after another and that period and hyphen cannot occur consecutively. This means that at no point in a file name should the character sequences **..**, **--**, **.-**, or **-.** appear. This file naming convention helps ensure that common vulnerabilities such as file names being too long, double hyphens being interpreted as a flag, or double periods being interpreted as the character sequence to ascend one level in the hierarchy of the filesystem.

More Validation

Validation of all inputted and outputted values should be occurring to ensure that they fall within an expected range of values. Inputted or outputted values that have not undergone validation pose the risk of allowing potentially malicious data to cause harm depending on where it is used and how it is operated upon. Whenever an input is taken or an output is given in a program, the range of expected values should be determined and then a method of validation that whitelists only those expected values should be implemented to ensure that only expected values are operated upon. Allowing malicious data in to be operated upon can be as benign as an exception being thrown and logged or as bad as malicious code being executed leading to potential damages. Of all recommendations given in this section this recommendation stands as being the highest priority as it provides the widest breadth of security and helps protect against what are potentially some of the most damaging exploits.

Works Cited

Mohindra, Dhruv, et al. “ERR01-J. Do Not Allow Exceptions to Expose Sensitive Information.” *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/ERR01-J.+Do+not+allow+exceptions+to+expose+sensitive+information.

Mohindra, Dhruv, et al. “FIO16-J. Canonicalize Path Names before Validating Them .” *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/FIO16-J.+Canonicalize+path+names+before+validating+them.

Mohindra, Dhruv, et al. “IDS01-J. Normalize Strings before Validating Them .” *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS01-J.+Normalize+strings+before+validating+them.

Mohindra, Dhruv, et al. “IDS51-J. Properly Encode or Escape Output.” *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS51-J.+Properly+encode+or+escape+output.

Svoboda, David, et al. “IDS50-J. Use Conservative File Naming Conventions.” *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS50-J.+Use+conservative+file+naming+conventions.

Appendix

Bug Name: Exposed Sensitive File System Information Through IO Exceptions
Bug ID: PR2-ERR01-01.

Assigned to: Alexander Marquis
Reported By: Douglas Lundin
Reported On: 1/28/2022

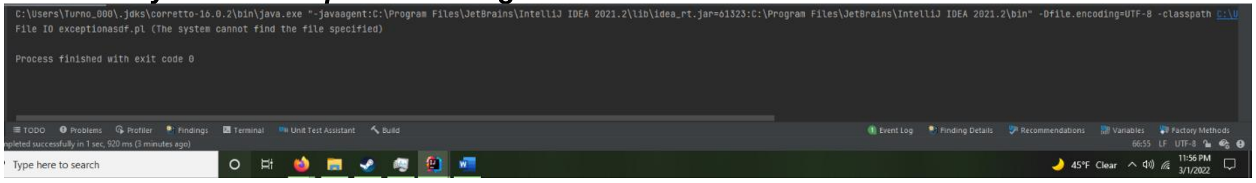
Severity: Medium[6]
Priority: P4[1]
Status: Closed

Area Path: CSC245-Project-2/src/CSC245_Project2.java
Environment: Windows 10/IntelliJ IDEA 2021.2 Ultimate Edition Build #IU-212.4746.92/Amazon Corretto v.16.0.2

Reason: Failure to properly comply with CERT code ERR01-J[1]
Description: IO exceptions suffered while finding or reading the file that the inputted file name points has the potential to expose sensitive filesystem information.
Steps to Reproduce:

1) Load *asdf.pl* as a program argument to program CSC245-Project-2.
2) Execute program CSC245-Project-2

Expected Result: Program prints to the console a terse message stating that the file that was seeking to be read was invalid. The user should not know why it was invalid, just that it was invalid (It was invalid because it either couldn't be found, opened, or it suffered some other IO exception).
Actual Result: Program prints to the console that a file IO exception has occurred followed by the *IOException* message.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61323:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
File IO exceptionasdf.pl (The system cannot find the file specified)

Process finished with exit code 0
```

[6] Mohindra, Dhruv, et al. "ERR01-J. Do Not Allow Exceptions to Expose Sensitive Information." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/ERR01-J.+Do+not+allow+exceptions+to+expose+sensitive+information.

Bug Name: Exposed Failure to Close File Through IO Exceptions
Bug ID: PR2-ERR01-02.

Assigned to: Alexander Marquis
Reported By: Douglas Lundin
Reported On: 1/28/2022

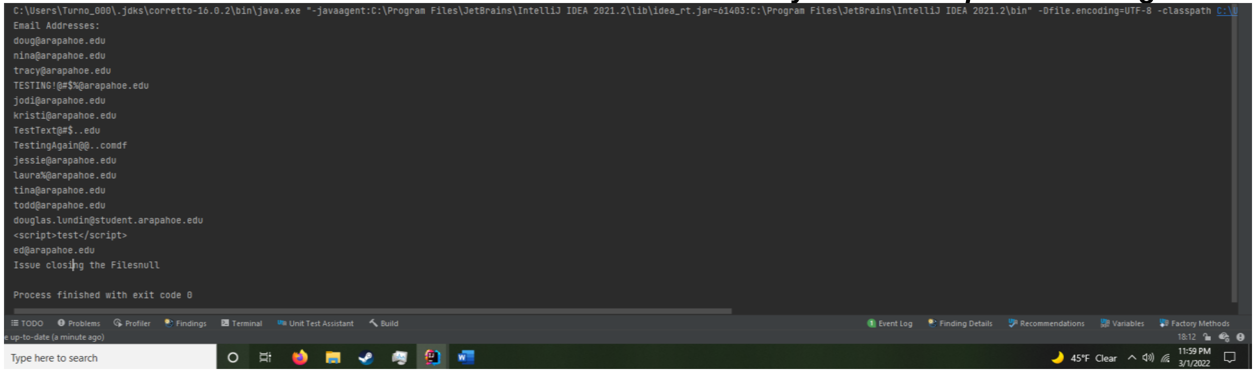
Severity: Medium[1]
Priority: P4[1]
Status: Closed

Area Path: CSC245-Project-2/src/CSC245_Project2.java
Environment: Windows 10/IntelliJ IDEA 2021.2 Ultimate Edition Build #IU-212.4746.92/Amazon Corretto v.16.0.2

Reason: Failure to properly comply with CERT code ERR01-J[1]
Description: IO exceptions suffered while attempting to close a file expose sensitive information about the file not being closed.
Steps to Reproduce:

1) Load standard program argument *Email_addresses_20210205.txt*.
2) Since actually failing to close a file is both difficult and dangerous, the *try* block that exists above the *catch* block that hosts this bug is modified to throw an *IOException* to trigger the catch block and the bug.
    a. The modification was the addition of a *throw new IOException();* statement at the top of the first try block and the temporary removal of the *if* block that normally occupies the *try* block
3) Execute program CSC245-Project-2 with modified try block

Expected Result: Program prints to the console a terse message stating that an operation failed. What operation failed should not be exposed to the user.
Actual Result: That the file has failed to close followed by the *IOException* message



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61403:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
Email Addresses:
doug@arapahoe.edu
nina@arapahoe.edu
tracy@arapahoe.edu
TESTING!@#$%@arapahoe.edu
jodi@arapahoe.edu
kristi@arapahoe.edu
TestText@#$..edu
TestingAgain@@..comdf
jessie@arapahoe.edu
laura%@arapahoe.edu
tina@arapahoe.edu
todd@arapahoe.edu
douglas.lundin@student.arapahoe.edu
<script>test</script>
ed@arapahoe.edu
Issue closing the Filesnull

Process finished with exit code 0
```

Bug Name: Failure to Normalize Program Argument Before Validation
Bug ID: PR2-IDS01-01.
Area Path: CSC245-Project-2/src/CSC245_Project2.java
Environment: Windows 10/IntelliJ IDEA 2021.2 Ultimate Edition Build #IU-212.4746.92/Amazon Corretto v.16.0.2

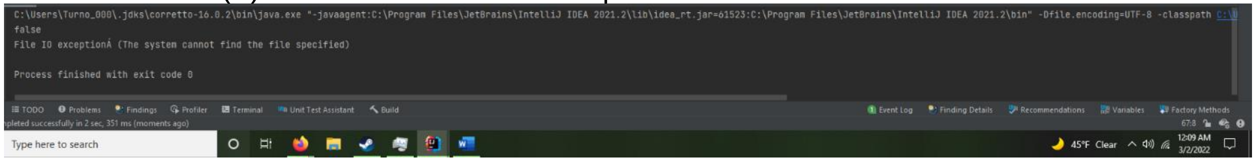
Assigned to: Alexander Marquis
Reported By: Douglas Lundin
Reported On: 1/28/2022

Reason: Failure to properly comply with CERT code IDS01-J[2]
Severity: Medium[7]
Priority: P12[2]
Status: Closed

Description: Failure to normalize program argument (inputted filename) before validation
Steps to Reproduce:

1) Load Á as a program argument.
2) Print if the putted argument is equal to \u0041\u0301 (its decomposed form)
3) Execute program CSC245-Project-2

Expected Result: Program normalizes the program argument representing a filename before its validation. In demonstration, the program should print out *true* representing that the A-acute character (Á) matches its decomposed form.
Actual Result: Program does not normalize program argument representing a filename before validation. In demonstration, the program prints out *false* representing that the A-acute character(Á) does not match its decomposed form.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61523:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\
false
File IO exceptionÁ (The system cannot find the file specified)

Process finished with exit code 0
```

[7] Mohindra, Dhruv, et al. "IDS01-J. Normalize Strings before Validating Them ." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS01-J.+Normalize+strings+before+validating+them.

Bug Name: Failure to Normalize File Contents Before Validation
Bug ID: PR2-IDS01-02.
Area Path: CSC245-Project-2/src/CSC245_Project2.java
Environment: Windows 10/IntelliJ IDEA 2021.2 Ultimate Edition Build #IU-212.4746.92/Amazon Corretto v.16.0.2

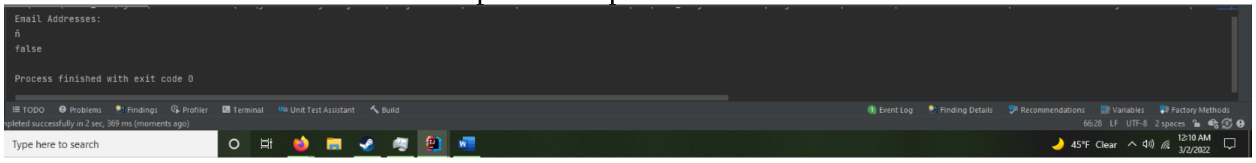
Assigned to: Alexander Marquis
Reported By: Douglas Lundin
Reported On: 1/28/2022

Reason: Failure to properly comply with CERT code IDS01-J[2]
Severity: Medium[2]
Priority: P4[2]
Status: Closed

Description: Failure to normalize input from file before validation.
Steps to Reproduce:

1) Load PR2_IDS01_02_Demonstration.txt as a program argument.
    a. This file has ñ as the only text
2) Add print statement just after character is printed to the console that prints whether the input is equal to its normalized form \u00f1
3) Execute program CSC245-Project-2

Expected Result: Program normalizes the contents of a file before its validated. In demonstration, the program should print out a series of *true* lines representing that the normalized characters match their expected output.
Actual Result: Program does not normalize the contents of a file before its validated. In demonstration, the program should print out a series of *false* lines representing that the normalized characters match their expected output



```
Email Addresses:
ñ
false

Process finished with exit code 0
```

Bug Name: Failure to Canonicalize Path Before Validation
Bug ID: PR2-FIO16-01.
Area Path: CSC245-Project-2/src/CSC245_Project2.java
Environment: Windows 10/IntelliJ IDEA 2021.2 Ultimate Edition Build #IU-212.4746.92/Amazon Corretto v.16.0.2

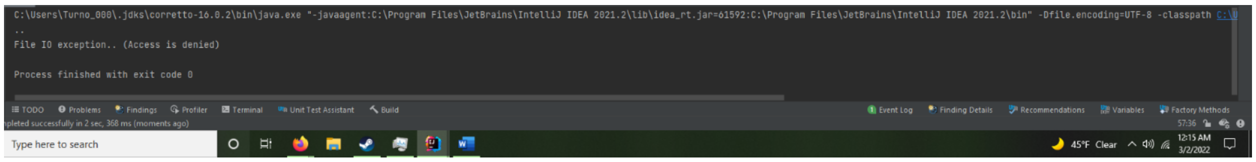
Assigned to: Alexander Marquis
Reported By: Douglas Lundin
Reported On: 1/28/2022

Reason: Failure to properly comply with CERT code FIO16-J[3]
Severity: Medium[8]
Priority: P4[3]
Status: Closed

Description: Failure to canonicalize path names and failure to validate that path name points at intended directory.
Steps to Reproduce:

1) Load .. as a program argument.
2) Print out the filename just prior to it being used.
3) Execute program CSC245-Project-2

Expected Result: Outside the demonstration the program should only display a message informing the user that the chosen file was invalid. With the demonstration active, before the message mentioned previously the program should also print out the canonicalized path so it can be inspected to ensure that it properly falls within the whitelisted directory
Actual Result: Outside demonstration the program suffers an IO exception as it tries to open and read the contents of a directory and prints its IO exception message (part of bug PR2-ERR01-01). With the demonstration active it the program prints out the non-canonicalized path name before the previously mentioned message.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61592:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
..
File IO exception.. (Access is denied)

Process finished with exit code 0
```

[8] Mohindra, Dhruv, et al. "FIO16-J. Canonicalize Path Names before Validating Them ." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/FIO16-J.+Canonicalize+path+names+before+validating+them.

Bug Name: Failure to Enforce Conservative File Naming Conventions
Bug ID: PR2-IDS50-01.
Area Path: CSC245-Project-2/src/CSC245_Project2.java
Environment: Windows 10/IntelliJ IDEA 2021.2 Ultimate Edition Build #IU-212.4746.92/Amazon Corretto v.16.0.2

Assigned to: Alexander Marquis
Reported By: Douglas Lundin
Reported On: 1/28/2022

Reason: Failure to properly comply with CERT code IDS50-J[4]
Severity: Medium[9]
Priority: P4[4]
Status: Closed

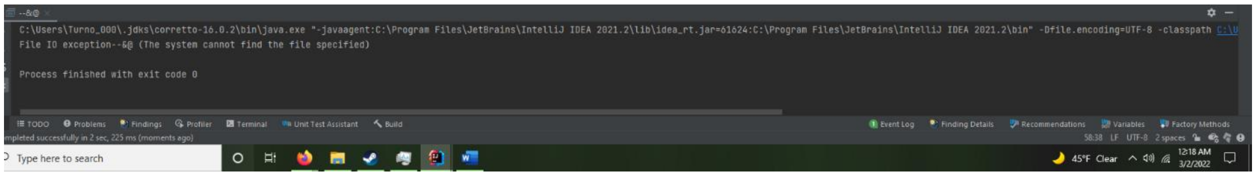
Description: Failure to only accept inputted file names from users that only allow for secure and regular character sequences. Current conservative file naming convention requires:

- File names shall be no shorter than two characters in length
- The first and last characters of the filename shall be alphanumeric
- The middle characters of the filename shall be alphanumeric, a period, a hyphen, or an underscore.
- A period shall not be directly followed by another period or a hyphen
- A hyphen shall not be directly followed by another hyphen or a period
- A filename shall not exceed 255 characters in length

Steps to Reproduce:

1) Load --&@. as a program argument.
2) Execute program CSC245-Project-2

Expected Result: Program prints out terse message stating that the file was invalid.
Actual Result: Program reads and prints the contents of the demonstration file named --&@ from secure directory to the console.



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61624:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
File IO exception--&@ (The system cannot find the file specified)

Process finished with exit code 0
```

[9] Svoboda, David, et al. "IDS50-J. Use Conservative File Naming Conventions." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS50-J.+Use+conservative+file+naming+conventions.

Bug Name: Failure to Properly Encode or Escape Output of File Contents
Bug ID: PR2-IDS51-01.
Area Path: CSC245-Project-2/src/CSC245_Project2.java
Environment: Windows 10/IntelliJ IDEA 2021.2 Ultimate Edition Build #IU-212.4746.92/Amazon Corretto v.16.0.2

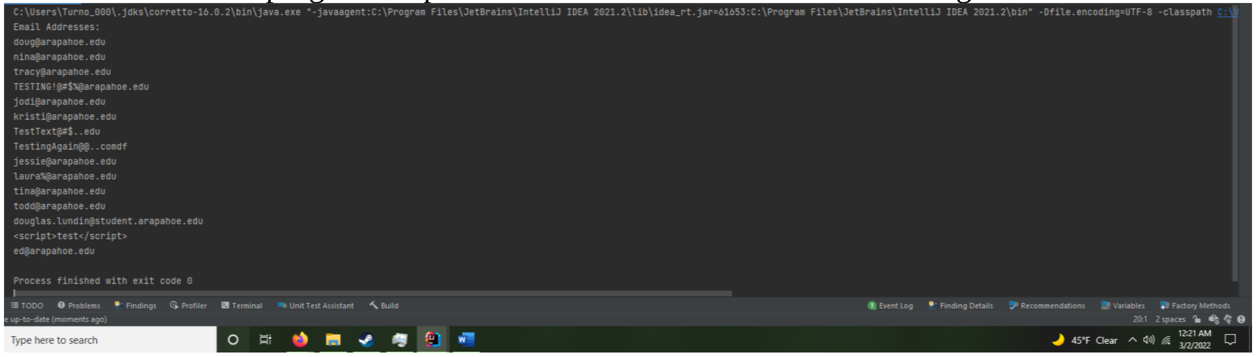
Assigned to: Alexander Marquis
Reported By: Douglas Lundin
Reported On: 1/28/2022

Reason: Failure to properly comply with CERT code IDS51-J[5]
Severity: Medium[10]
Priority: P4[5]
Status: Closed

Description: Failure to ensure that the printed output, the output being the read file's contents, is securely encoded, escaped, or overall sanitized.
Steps to Reproduce:

1) Load the standard program argument of Email_addresses_20210205.txt as a program argument.
2) Execute program CSC245-Project-2

Expected Result: The program outputs text with dangerous characters encoded and all script tags removed.
Actual Result: The program outputs the full file contents with no encoding or sanitization



```
C:\Users\Turno_000\.jdks\corretto-16.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\lib\idea_rt.jar=61653:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath C:\U
Email Addresses:
doug@arapahoe.edu
nina@arapahoe.edu
tracy@arapahoe.edu
TESTING!@#$%@arapahoe.edu
jodi@arapahoe.edu
kristi@arapahoe.edu
TestText@#$..edu
TestingAgain@@..comdf
jessie@arapahoe.edu
laura%@arapahoe.edu
tina@arapahoe.edu
todd@arapahoe.edu
douglas.lundin@student.arapahoe.edu
<script>test</script>
ed@arapahoe.edu

Process finished with exit code 0
```

[10] Mohindra, Dhruv, et al. "IDS51-J. Properly Encode or Escape Output." *Confluence*, Carnegie Mellon University Software Engineering Institute, https://wiki.sei.cmu.edu/confluence/display/java/IDS51-J.+Properly+encode+or+escape+output.