# Path Manipulation Development Mitigation SOP

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

* An attacker can specify a path used in an operation on the file system.
* By specifying the resource, the attacker gains a capability that would not otherwise be permitted.

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

## Defense Against Path Manipulation

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

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 However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date.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

## Example

public ImporterTask getImporter(PDFViewer viewer, URL url)  
 throws IOException {  
 if (url.getProtocol().equals("file")) {  
 File file;  
 try {  
 file = new File(new URI(url.toString()));  
 } catch (URISyntaxException e) {  
 file = new File(url.getPath());  
 }  
 if (file != null) {  
 return getImporter(file);  
 }  
 }  
 this.url = url;  
 return getImporter(url);  
 }

## Explanation

The attacker can specify the value that enters the program at the getParameter() function in PDFViewerApplet.java, and this value is used to access a file system resource at File() in ProjPDFImporter.java.

## Recommendation

public ImporterTask getImporter(PDFViewer viewer, URL url)throws IOException{  
  
// The file protocol will never be used as documents are saved as urls and translated to GUIDS.  
// Show the user an error if this unlikely case is ever hit.  
 if (url.getProtocol().equals("file")){  
 throw new IOException("Unsupported import action. Please report this error.");  
 } else {  
 this.url = url;  
 return getImporter(url);  
 }  
 }

## SonarSource

User provided data, such as URL parameters, POST data payloads, or cookies, should always be considered untrusted and tainted. Constructing file system paths directly from tainted data could enable an attacker to inject specially crafted values, such as '../', that change the initial path and, when accessed, resolve to a path on the filesystem where the user should normally not have access.

A successful attack might give an attacker the ability to read, modify, or delete sensitive information from the file system and sometimes even execute arbitrary operating system commands. This is often referred to as a "path traversal" or "directory traversal" attack.

The mitigation strategy should be based on the whitelisting of allowed paths or characters.

**Recommendation**

1. <https://vulncat.fortify.com/en/detail?id=desc.dataflow.java.path_manipulation>
2. <https://rules.sonarsource.com/java/RSPEC-2083>
3. <https://cwe.mitre.org/data/definitions/23>
4. <https://cwe.mitre.org/data/definitions/36>