**System Information Leak: External Development Mitigation SOP**

An external information leak occurs when system data or debugging information leaves the program to a remote machine via a socket or network connection. Any information that can expose specifics to a remote user about the operating systems used, full pathnames, other users, locations of configuration files or any information regarding internal resources. In some cases the error message tells the attacker precisely what sort of an attack the system will be vulnerable to.

**Defense against System Information Leak: External**

Most information leaks are divulged through error messages. Keep security in mind when writing error messages to be displayed on the screen. In production environments, keep error messages as brief as possible. The GUID is the only recommended information to be provided to the user to help them diagnosis their issue.

**Example**

1. String exceptionId = ErrorUtil.generateUniqueId();
2. model.addAttribute("exceptionIncidentId", exceptionId);
3. String errorMsg = "Document Id: " + documentId + " may be invalid.";
4. model.addAttribute("exceptionMessage", errorMsg);
5. ErrorUtil.log(exceptionId, errorMsg , exception);
6. }

**Explanation**

1. The example will display the documentID to the user from the exceptionMessage on their screen.
2. Internal information should never be exposed to the user.
3. Rely on the standard error message generated by the application in the production environment.

**Recommendation**

1. String exceptionId = ErrorUtil.generateUniqueId();
2. model.addAttribute("exceptionIncidentId", exceptionId);
3. String errorMsg = "Document Id: " + documentId + " may be invalid.";
4. ErrorUtil.log(exceptionId, errorMsg , exception);
5. }

References

[1] [OWASP – Information Leak](https://www.owasp.org/index.php/Information_Leakage)

[2] [System Information Leak: External](http://www.hpenterprisesecurity.com/vulncat/en/vulncat/cpp/system_information_leak_external.html)